Anomaly Detection, Prognostics, and Diagnostics: Machine Learning for the Hadron Calorimeter at the CMS Experiment

Mulugeta Weldezgina Asres

UiA University of Agder

Doctoral Dissertations at
the University of Agder 477

Mulugeta Weldezgina Asres

Anomaly Detection, Prognostics, and Diagnostics: Machine Learning for the Hadron Calorimeter at the CMS Experiment

Doctoral Dissertation for the Degree *Philosophiae Doctor (Ph.D.)*
at the Faculty of Engineering and Science, Specialisation in ICT: Artificial Intelligence

University of Agder
Faculty of Engineering and Science
2024

Doctoral Dissertations at the University of Agder 477
ISSN: 1504-9272
ISBN: 978-82-8427-194-1

Printed by Make!Graphics
Kristiansand

Preface

The Large Hadron Collider (LHC) is the most powerful particle collider ever built worldwide, providing thousands of researchers opportunities for groundbreaking discoveries. The LHC accelerates proton and heavy ions to speeds approaching the speed of light—300 thousand kilometers per second—and collides them to reveal new physics and answer the grandest questions of our universe. The scientific discoveries of the LHC and technological advancements at CERN have delivered tremendous contributions to our daily lives; some of the outstanding examples are the World Wide Web, touchscreen technology, hadron tumor therapy, and PET scanners for medical imaging. The Compact Muon Solenoid (CMS) experiment is one of the two general-purpose particle physics detectors at the LHC and it explores a broad range of particle physics, like the search for the Higgs boson, extra dimensions, and dark matter. The discovery and characterization of the Higgs boson, which is among the distinguished contributions of the data collected by the CMS experiment, is the latest major discovery in physics and adds a piece to the puzzle of the exciting world of subatomic particles. The particle collisions at the LHC produce an enormous amount of data at the energy and intensity frontiers of high-energy physics, and the CMS detector employs numerous sensors to acquire the data at high data rates. The detector has recently undergone crucial upgrades—leveraging new technologies to achieve the high luminosity LHC program; its components are growing more complex to support high radiation exposure, strong magnetic fields, and remarkable particle acquisition rates. Such extraordinary attributes pose tremendous challenges for the experiment and data processing. Ensuring the quality of physics data requires timely monitoring and the resolution of detector anomalies. Recent advancements in machine learning tools demonstrate promise in addressing high-energy physics challenges, such as detector simulation, real-time analysis and triggering, particle event reconstruction and identification, calibration, detector monitoring, and preemptive maintenance. Our study presents deep learning models for automated monitoring of the Hadron Calorimeter (HCAL) of the CMS experiment using high-dimensional diagnostics and data quality monitoring data sets.

Acknowledgments

The road of our study was full of challenges due to the complexity of LHC systems, high-dimensional and large data sets, the COVID lockdown and remote work, and the unfortunate Ukraine war. The study required tremendous coordination, close collaboration with several experts, and unique solution development. The experience has provided us with exciting academic and professional growth opportunities.

First and foremost, my most profound appreciation goes to my Ph.D. supervisor—Prof. Christian Omlin—for allowing me to pursue the study, for his guidance in overcoming difficulties, and for solid mentorship and support throughout the study.

Second, we are deeply grateful to Prof. Jay Dittmann, Baylor University—the former project manager of the CMS-HCAL—for his close follow-up and guidance throughout the study, candid knowledge sharing on the HCAL systems, collaboration in gathering the sensor data sets, analysis of our results, and participation in editing our manuscripts. We also express our gratitude to Alberto Belloni—the succeeding manager—for continuing and fostering our collaboration with the HCAL.

Third, we appreciate the HCAL operation and detector performance groups for their invaluable insights on the HCAL diagnostics sensors and data quality monitoring (DQM). A big thanks go to Pavel Parygin, Grace Cummings, Long Wang, David Yu, Maria Toms, Alan Campbell, Seth I. Cooper, and Aleko Khukhunaishvili for their close cooperation in providing expert views on the HCAL systems and for scrutinizing our solutions. We further thank Long Wang for his support during the integration of our DQM models into the CMSSW production systems. We appreciate the DQM and machine learning groups of CMS for coordinating forum sessions—enabling us to present our work to wider field expert audiences—and reviewing our manuscripts. Finally, our gratitude goes to the HCAL publication committee for facilitating the publication approval process and reviewing our manuscripts.

The most harrowing and excruciating moment during our study was the trauma of not knowing my family's whereabouts for almost two years in my home country, Ethiopia, due to the gruesome genocidal war since Nov 2020 on the people of Tigray. Over a million people reportedly perished from the war and systematic starvation from a two-year-long utter siege. I sincerely appreciate the support of my supervisor and friends during those unbearable dark times. I am also deeply thankful to my spouse and kids for their unwavering support throughout the long working hours.

Mulugeta Weldezgina Asres
University of Agder, Grimstad, Norway

Abstract

Machine Learning (ML) tools have gained immense popularity due to the proliferation of sensor data for monitoring, prognostic, and diagnostic applications in various industrial domains. The growing system complexity and monitoring data volumes of the Large Hadron Collider (LHC) at CERN accentuates the need for automation through advanced ML tools. Detection, identification, and resolution of anomalies are essential to generate more physics collision data of the highest quality. Developing ML tools for complex systems often involves expensive data curation and modeling efforts; it requires adequate, cleaned, and annotated data sets, and addresses the challenges of heterogeneity and curse-of-dimensionality of large data sets. The Compact Muon Solenoid (CMS) experiment—one of the large general-purpose colliders at the LHC—has dedicated substantial monitoring efforts for detector systems and particle data quality; the control and safety systems (DCS/DSS) actively monitor safety-critical problems, and the data quality monitoring (DQM) system mitigates data loss by identifying and diagnosing physics data problems. The existing monitoring systems need to incorporate a wide range of monitoring variables and adapt to the evolving conditions of the detectors. This dissertation focuses on the development of unsupervised anomaly detection (AD), anomaly prediction (AP), and root-cause analysis (RCA) on multivariate time series data sets. We have developed deep learning models for frontend electronics of the Hadron Calorimeter (HCAL) of the CMS detector using diagnostic sensors and high-dimensional particle acquisition channel-monitoring data sets. We have employed subsystem-granularity modeling using a divide-and-conquer approach to monitor the complex HCAL systems with thousands of sensors. Our monitoring tools have detected and identified previously unknown and hard-to-monitor anomalies, and extended the monitoring, diagnostics, and prognostics automation of the HCAL. The developed tools are deployed at CERN and are currently providing essential real-time and offline anomaly monitoring and diagnostics on the frontend electronics of the HCAL and the online DQM system. Our scientific contribution in tackling the challenges for complex system monitoring includes: 1) enhancing multivariate sensor AD, 2) a promising AP approach, 3) context-aware high-dimensional spatio-temporal AD, 4) transfer learning on multi-network deep learning models, 5) lightweight interconnection and divergence discovery for multi-systems with multivariate sensors, and 6) enhancing computational efficiency of anomalies causality discovery on binary anomaly data.

Sammendrag

Maskinlæringsverktøy (ML) har fått enorm popularitet på grunn av spredning av sensordata for overvåking, prognostiske og diagnostiske applikasjoner i ulike industrielle domener. Den økende systemkompleksiteten og overvåkingen av datamengdene til Large Hadron Collider (LHC) ved CERN understreker behovet for automatisering gjennom avanserte maskinlæringsverktøy. Deteksjon, identifisering og oppløsning av uregelmessigheter er avgjørende for å generere flere fysiske kollisjonsdata av høyeste kvalitet. Utvikling av maskinlæringsverktøy for komplekse systemer innebærer ofte kostbar datakurering og modellering. Det krever tilstrekkelige, rensede og kommenterte datasett, og adresserer utfordringene med heterogenitet og forbannelse av dimensjonalitet av store dataset. Compact Muon Solenoid (CMS) eksperimentet—en av de store generelle kolliderne ved LHC—har dedikert betydelig overvåkingsinnsats for detektorsystemer og partikkeldatakvalitet; kontroll- og sikkerhetssystemene (DCS/DSS) overvåker aktivt sikkerhetskritiske problemer, og datakvalitetsovervåkingssystemet (DQM) reduserer datatap ved å identifisere og diagnostisere fysikkdataproblemer. De eksisterende overvåkingssystemene må inkludere et bredt spekter av overvåkingsvariabler og tilpasse seg de utviklende forholdene til detektorene. Denne avhandlingen fokuserer på utvikling av uovervåket anomalideteksjon (AD), anomalidediksjon (AP) og rotårsaksanalyse (RCA) på multivariate tidsseriedatasett. Vi har utviklet dype læringsmodeller for frontend-elektronikk av Hadron Calorimeter (HCAL) av CMS detektoren ved hjelp av diagnostiske sensorer og høydimensjonale partikkelinnsamlingskanalovervåkingsdatasett. Vi har brukt delsystem-granularitetsmodellering ved hjelp av en splitt-og-hersk-tilnærming for å overvåke de komplekse HCAL-systemene med tusenvis av sensorer. Våre overvåkingsverktøy har oppdaget og identifisert tidligere ukjente uregelmessigheter som er vanskelige å overvåke, og utvidet overvåking, diagnostikk og prognoseautomatisering av HCAL. De utviklede verktøyene er distribuert på CERN og gir for tiden viktig sanntids og offline avviksovervåking og diagnostikk på frontend-elektronikken til HCAL og online DQM-systemet. Vårt vitenskapelige bidrag til å takle utfordringene for kompleks systemovervåking inkluderer: 1) forbedring av multivariat sensor AD, 2) en lovende AP-tilnærming, 3) kontekstbevisst høydimensjonal romlig-temporal AD, 4) overføringslæring på multi-nettverk dype læringsmodeller, 5) lett sammenkobling og divergensoppdagelse for multisystemer med multivariate sensorer, og 6) forbedring av beregningseffektiviteten av anomalies kausalitetsoppdagelse på binære anomalidata.

Publications

- **Paper-1, Conference**: Mulugeta W. Asres, Grace Cummings, Pavel Parygin, Aleko Khukhunaishvili, Maria Toms, Alan Campbell, Seth I. Cooper, David Yu, Jay Dittmann, and Christian W. Omlin. "Unsupervised Deep Variational Model for Multivariate Sensor Anomaly Detection," in IEEE International Conference on Progress in Informatics and Computing (PIC), pp. 364-371. IEEE, 2021. doi: 10.1109/PIC53636.2021.9687034

- **Paper-2, Conference**: Mulugeta W. Asres, Grace Cummings, Aleko Khukhunaishvili, Pavel Parygin, Seth I. Cooper, David Yu, Jay Dittmann, and Christian W. Omlin. "Long Horizon Anomaly Prediction in Multivariate Time Series with Causal Autoencoders," in PHM Society European Conference, vol. 7, no. 1, pp. 21-31, 2022. doi: 10.36001/phme.2022.v7i1.3367

- **Paper-3, Workshop**: Mulugeta W. Asres, Long Wang, David Yu, and Christian W. Omlin. "Spatio-Temporal Anomaly Detection for the DQM of the CMS Experiment via Graph Networks," 5th Inter-experiment Machine Learning Workshop, CERN, 2022.

- **Paper-4, Journal**: Mulugeta W. Asres, Christian W. Omlin, Long Wang, David Yu, Pavel Parygin, Jay Dittmann, Georgia Karapostoli, Markus Seidel, Rosamaria Venditti, Luka Lambrecht, Emanuele Usai, Muhammad Ahmad, Javier Fernandez Menendez, Kaori Maeshima, and the CMS-HCAL Collaboration. "Spatio-Temporal Anomaly Detection with Graph Networks for Data Quality Monitoring of the Hadron Calorimeter," in Sensors, vol. 23, no. 24, pp. 9679, MDPI, 2023, doi: 10.3390/s23249679.

- **Paper-5*, Journal**: Mulugeta W. Asres, Christian W. Omlin, Long Wang, Jay Dittmann, and the CMS-HCAL Collaboration. "Extending Anomaly Detection for Data Quality Monitoring through Transfer Learning between Calorimeters," CERN Detector Notes, 2023, doi: DN2023/023. (*Waiting for the CMS-HCAL authorship list preparation approval*).

- **Paper-6, Conference**: Mulugeta W. Asres, Christian W. Omlin, Jay Dittmann, Pavel Parygin, Joshua Hiltbrand, Seth I Cooper, Grace Cummings, and David Yu. "Lightweight Multi-System Multivariate Interconnection and Divergence Discovery," arXiv preprint, 2024, doi: 10.48550/arXiv.2404.08453. (*Accepted in 19th Annual System of Systems Engineering Conference, IEEE*).

- **Paper-7*-Journal**: Mulugeta W. Asres, Christian W. Omlin, Jay Dittmann, Pavel Parygin, and the CMS-HCAL Collaboration. "Scalable Online Anomaly Causality Discovery for Large Complex Systems," CERN Detector Notes, 2023, doi: CMS DN-2023/030. (*Waiting for the CMS-HCAL authorship list preparation approval*).

Contents

List of Figures

List of Tables

List of Abbreviations

CERN Abbreviations	
BE	Back end of the HCAL
BSM	Beyond the Standard Model
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
CU	Calibration Unit
DAQ	Data Acquisition System
DQM	Data Quality Monitoring
FE	Front end of the HCAL
HCAL	Hadron Calorimeter
HB	HCAL Barrel Subdetector
HBHE	HCAL Barrel and Endcap Subdetectors
HCAL	Hadron Calorimeter
HE	HCAL Endcap Subdetector
HEM	HE Minus Hemisphere RBX
HEP	HE Plus Hemisphere RBX
HLT	High-Level Trigger System
HPD	Hybrid Photodiode Transducer
L1	Level 1 Trigger System
LHC	Large Hadron Collider
LS	Lumisection
ngCCM	Next-generation Clock, Control, and Monitoring Module
P5	Point 5 Location Site at the LHC Ring
QIE	Charge Integrator and Encoder
RBX	Readout Box
RM	Readout Module
RunID	Collision Run Identification Number
SiPM	Silicon Photomultiplier
SM	Standard Model

Non-CERN Abbreviations	
1D/2D/3D	One/Two/Three Dimensional
AD	Anomaly Detection
AE	Autoencoder
ANAC	Our Anomaly Aware CI Test
AnomalyCD	Our Anomaly Causality Discovery Approach
AnoP	Our Forecast-based Anomaly Prediction Model
AP	Anomaly Prediction
BN	Bayesian Network
DAG	Directed Acyclic Graph
CAM	Class Activation Map
CD	Causality Discovery
CGVAE	Our CNN, GRU, and VAE-based AD Model
CI	Conditional Independence
CNN	Convolutional Neural Networks
CPD	Conditional Probability Distribution
DAD	Deep Anomaly Detection
DESMOD	Our Detector System Monitoring and Diagnostics Project
DL	Deep Learning
FA	Feature Attribution
FC	Fully-Connected Neural Networks
GAN	Generative Adversarial Networks
GCM	Graph Causal Model
GCN	Graph Convolutional Neural Networks
GNN	Graph Neural Networks
GRU	Gated Recurrent Unit Networks
GraphSTAD	Graph-Based Spatio-Temporal AD Model
IG	Integrated Gradient
KL	Kullback-Leibler Divergence
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MD	Mahalanobis Distance
ML	Machine Learning
MSE	Mean Squared Error
MTS	Multivariate Time Series
PCA	Principal Component Analysis
PdM	Predictive Maintenance
RCA	Root-Cause Analysis
ResNet	Residual Neural Networks
RNN	Recurrent Neural Networks
S2S	Sequence-to-Sequence
SHAP	Shapley Additive Explanations
SSM	State-Space Model
ST	Spatio-Temporal

TCN	Temporal Convolutional Neural Networks
TS	Time Series
TSAD	Time Series Anomaly Detection
TSC	Time Series Classification
TSF	Time Series Forecasting
TSR	Time Series Regression
UTS	Univariate Time Series
VAE	Variational Autoencoder

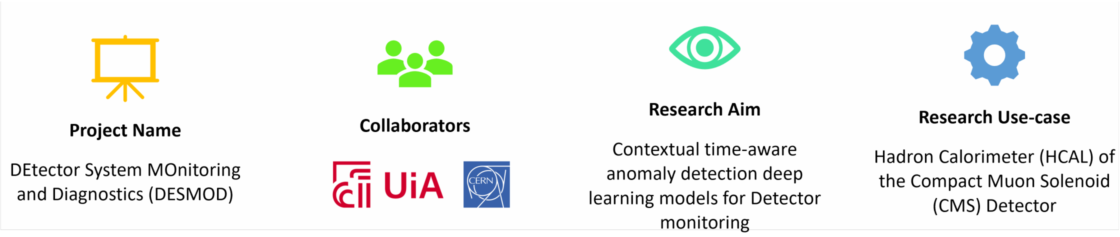
Chapter 1

Introduction

This research is part of the collaboration between UiA and CMS experiment under the project name of **DE**tector **S**ystem **MO**nitoring and **D**iagnostics (**DESMOD**), which focuses on artificial intelligence (AI) development for particle detector monitoring (see Fig. 1.1).

DESMOD aims to rapidly detect anomalies at the *Hadron Calorimeter* (HCAL) of the CMS experiment at the *Large Hadron Collider* (LHC) through contextual time-aware AD models for detector monitoring; it consists of ML tasks for diagnostics sensors, data quality monitoring, and fault causal discovery (see Fig. 1.2).

This chapter discusses the background and motivation of the study, highlights the research questions and the general methodology of the study, and provides the thesis roadmap.



Project Name

DEtector System MOnitoring and Diagnostics (DESMOD)

Collaborators

UiA

Research Aim

Contextual time-aware anomaly detection deep learning models for Detector monitoring

Research Use-case

Hadron Calorimeter (HCAL) of the Compact Muon Solenoid (CMS) Detector

Figure 1.1: Ph.D. study collaboration and research focus.

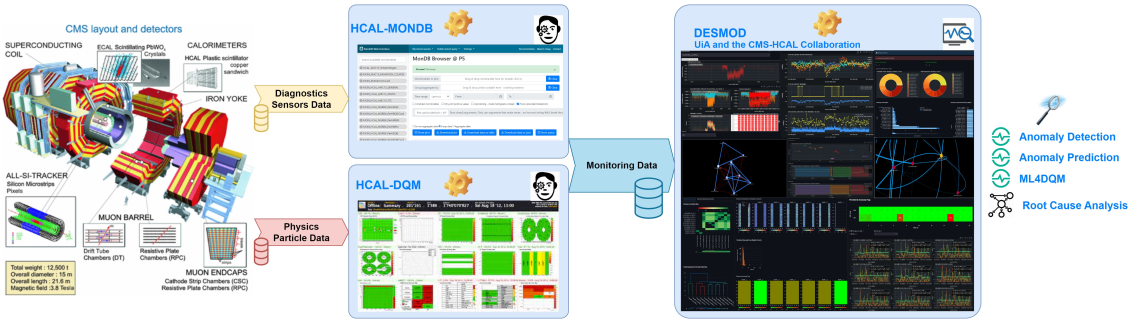


Figure 1.2: Detector system monitoring and diagnostics system (DESMOD).

1.1 Machine Learning in High-Energy Physics

This section briefly discusses ML development for high-energy physics applications at the LHC.

Physicists are searching for experimental evidence of particle physics—fundamental constituents of matter and the forces of nature—from the high-energy collision data of the LHC. While the *standard model* (SM) of particle physics has been extremely successful in describing the experimental data accumulated so far, many open questions remain for *physics beyond the SM* (BSM). The BSM searches for new rare phenomena that the adopted strategy for SM might miss; it is a key aspect of the physics program of present and future colliders [21]. The experimental high-energy physics program revolves around two main objectives: probing the SM with higher precision and searching for new physics [22].

The high-energy physics field is benefiting from advances in machine learning, but the complexity of the field demands unique solutions with further research and development [23]. The incorporation of ML in high-energy physics workflows is a fast-growing field and is expected to significantly improve tasks—including particle reconstruction and analysis, event simulation, pattern recognition and calibration, data compression, and real-time processing [22, 23]. The end-to-end learning capability of the recent deep learning method on high-energy physics has far-reaching leverage. Since low-level collision data tend to be high-dimensional and sparse, most high-energy physics analyses depend on high-level features curated by extensive domain knowledge from the collision data [22]. Deep learning offers end-to-end feature extraction from raw data—extracting high-level features without much reliance on domain knowledge. This can be beneficial for high-energy physics learning, with new opportunities to explore closely the raw data in LHC analyses. Recent advances in ML tools show promise in high-energy physics related to physics analysis study and the LHC performance [21–23].

ML tools operate either by enhancing the existing techniques such as the matrix element (ME) method [22, 24] or employ generic approaches like anomaly detection [21, 25–33] when searching rare physics in the high-energy physics. The ME is a popular technique for measuring physical model parameters and searching for new phenomena, but its computational cost limits its applicability. The LHC has used it to measure the Higgs and top quark of the SM [22]. The dramatic speed gain of modern ML can potentially address the limitation of ME and enable a greater reach for new physics discovery on the high luminosity LHC data [24]. The AD approaches formulate the searching for rare physics as catching unusual anomalies, where anomalies manifest as population particles detected collectively or individually [21]. Several studies recognize the high-efficiency potential of AD using deep learning algorithms for new physics searches [25–33]. While un(semi-)supervised approaches using autoencoder (AE) [27–30], variational AE (VAE) [31–33], and generative adversarial networks (GAN) [26] are popular in AD for searching for new physics, weakly supervised approaches with self-supervised learning are also gaining attention for performance improvement [34]. Refs. [34] and [29] propose transformer and graph neural networks (GNN), respectively, to improve the learning mechanisms of AD models for high-energy physics. Refs. [35,36] explore quantum machine learning (QML) as a candidate future tool for analyzing high-energy physics data. Although Ref. [36] claims there is no sufficient evidence that the current state of QML provides an advantage over classical ML, Ref. [35] presents AD using a quantum AE that outperforms classical AE—in accuracy and training efficiency—for resonant heavy Higgs signals detection.

The performance at the LHC is enhanced by leveraging ML models in several applications: 1) detector response and physics simulation, 2) real-time analysis and triggering, 3) particle event reconstruction and identification, 4) pile-up mitigation, and 5) detector monitoring [22,23,37–41]:

- *Detector response and physics simulation*: Scientists compare detector response data to the SM or new physics models to discover particles. Analytically computing detector response is impossible, so Monte Carlo simulation tools like GEANT [42] and PYTHIA [43] are being utilized. The simulators capture the relevant physics on a hierarchy of scales, starting with the microscopic interactions within a proton-proton collision and ending with the interaction of particles in the enormous LHC detectors [44]. Trillions of simulated collisions are needed for precision hypothesis testing, but such simulations are expensive [37]. High-fidelity fast generative models like GANs and VAEs offer a promising alternative by learning from existing data samples [24,45,46].

- *Real-time analysis and triggering*: The CMS trigger systems analyze the particle physics data in realtime to select interesting events with reasonable efficiency and distribute them for later offline physics study. Triggering starts the physics event selection process at a rate of 40 MHz. The growing high luminosity of the LHC poses a challenge for real-time triggering data analysis; machine learning offers the possibility of performing real-time analysis while improving the efficiency of triggering [47].

- *Event reconstruction and particle identification*: The physical processes of interest in high-energy physics experiments occur on time scales too short to be observed directly by particle detectors. To obtain a more accurate reconstruction and identification of the physical process, algorithms convert raw measurements from detector electronics into higher-level data objects, which correspond to the detected physical particles. Machine learning algorithms are applied at various steps of the reconstruction process, such as feature extraction, pattern recognition, object characterization, and multiple calorimeter combined reconstruction and particle identification [48].

- *Pileup mitigation*: A particle bunch crossing at the LHC has multiple nearly simultaneous proton-proton collisions. The collisions create a low transverse-momentum noise (pileup) to target high transverse-momentum events. The pileup affects the reconstruction accuracy of many physics observables. Pileup mitigation becomes a key ingredient of online and offline event reconstruction; ML models are proposed to improve pileup rejection performances [38].

- *Detector monitoring*: ML-based anomaly detection models are useful to learn from data and produce an alert for deviation caused by previously known or unknown problems. AD algorithms allow monitoring of a larger number of variables and can aid preemptive maintenance when sensitive to subtle signs forewarning of imminent failure [39–41, 49–51].

Several deep learning applications have been explored for—including but not limited to classification, regression, forecasting, and anomaly detection tasks. The applications employ models, such as deep neural network (DNN), convolutional (CNN), recurrent (RNN), graph (GNN), transformer, diffusion models, and normalizing flows. The models follow architectures like autoencoder (AE), variational AE (VAE), and GAN; trained using supervised, unsupervised, semi-supervised, reinforcement learning, and quantum machine learning mechanisms. Chapter 3 will provide a technical discussion of the methods; we limit our discussion in this section to AD applications in high-energy physics. We refer readers to the survey studies in Refs. [22, 37, 44] for a comprehensive discussion on the drivers and challenges of ML for high-energy physics. Ref. [22] presents promising future research and development areas with a roadmap for their implementation, and Ref. [37] provides a comprehensive list of citations of ML approaches to experimental, phenomenological, and theoretical analyses in high-energy physics.

1.1.1 Anomaly Detection for LHC Monitoring

Anomaly detection (AD) methods are employed in high-energy physics to identify abnormal events in the search for new physics phenomena beyond the SM and to capture faults in colliders [37]. Researchers are actively exploring AD for monitoring various LHC systems, such as the accelerator, detector, and data quality monitoring [39,40,49–54]. Refs. [53] and [54] propose an automated detection for anomalous behavior of the injection kicker magnets of the LHC accelerator using the Gaussian mixture model (GMM) and isolation forest on 2015-2016 and 2016-2019 data sets, respectively. The anomalies are predicted from time-window segments using moving average [53], and Fourier features [54]. Refs. [52, 55, 56] employ LSTM and GRU models to detect quenches and power abort anomalies on the LHC superconducting magnets. The authors propose using data binned by an adaptive signal quantizer to reduce computation latency. They utilized 2500 voltage and current data samples acquired during quench time. Ref. [39] proposes CNN AD using the rechit occupancy map data of the DQM for the ECAL and HCAL—moving from a rules-based assessment towards supervised and unsupervised ML models. The authors employed 2016-2017 data sets with 40K good samples and 8K artificially generated anomalies. Ref. [40] presents CNN and DNN models for monitoring drift tube chambers of the muon spectrometer of the CMS detector using occupancy map data. The authors have assessed the misbehavior of drift tubes with high granularity and combined the results to probe different detector components. They employ image-based feature extraction on a dataset containing 21K samples of 2D occupancy histograms from 250 chambers across 84 runs from LHC Run-2 collision data. Ref. [51] has investigated AD for the trigger system of the CMS detector using a conditional VAE (CVAE) model. The authors define a modified loss function that allows the CVAE to learn the optimal reconstruction resolution and designed a Kullback-Leibler (KL) divergence anomaly metric for improved performance. They utilize event paths data sets from the trigger L1 and HLT that contain 100K samples with 3% anomalies. A deep autoencoder has detected anomalies in JetHT particle data collected in the 2016 LHC collision run by the CMS experiment in Ref. [50]. The model achieves a promising low false-positive rate on 160K samples. The authors employ a feature selection approach that mimics the logic of the CMS procedure—utilizing the statistical features of collision data. Developing machine learning models using the DQM monitoring quantities comes with some impediments, such as data normalization to handle variation in experimental settings, the granularity of the failures to spot, and limited availability of the ground truth labels [40]. Only a few employed temporal end-to-end DL models on spatial data [39, 40].

1.2 The CMS Detector System Overview

This chapter describes the use-case system of our study—the HCAL of the CMS experiment. We will first briefly explain the LHC and the CMS experiment and then describe the HCAL systems. Our study focuses on the *readout boxes* (RBXes)—the frontend electronics of the HCAL—and we will discuss the operation and major system components of the RBXes—illustrated in Fig. 1.3.

The LHC is the largest energy particle collider ever built worldwide that commenced operation in 2008. It is designed to conduct experiments in physics and increase our understanding of the universe, with the expectation that new findings will lead to practical applications. The aim is to reveal BSM physics with the center of mass proton-proton collision energies of up to 14 TeV and nominal luminosity of 10^{34} $\text{cm}^{-2}\text{s}^{-1}$ [57, 58]. It collides heavy lead ions with an energy of 2.8 TeV per nucleon and a peak luminosity of 10^{27} $\text{cm}^{-2}\text{s}^{-1}$ [58].

The LHC consists of a 27-kilometer ring tunnel located 100 meters underground at the France-Switzerland border near Geneva [57]. The LHC is a two-ring superconducting hadron accelerator and collider capable of accelerating and colliding beams of protons and heavy ions with unprecedented luminosity at a velocity close to the speed of light—3×10^8 ms^{-1} (see Fig. 1.4). The protons and the heavy ions of the collision experiments are generated from hydrogen gas and predominantly solid lead metal, respectively, and form a plasma. The plasma passes through several small boosters and accelerators to speed up before reaching the LHC rings. The *super proton synchrotron* (SPS) accelerates the plasma and injects beams into the LHC in opposite directions—one beam traveling clockwise and the other going counterclockwise. Inside the LHC, the beams continue to accelerate up to the speed of light

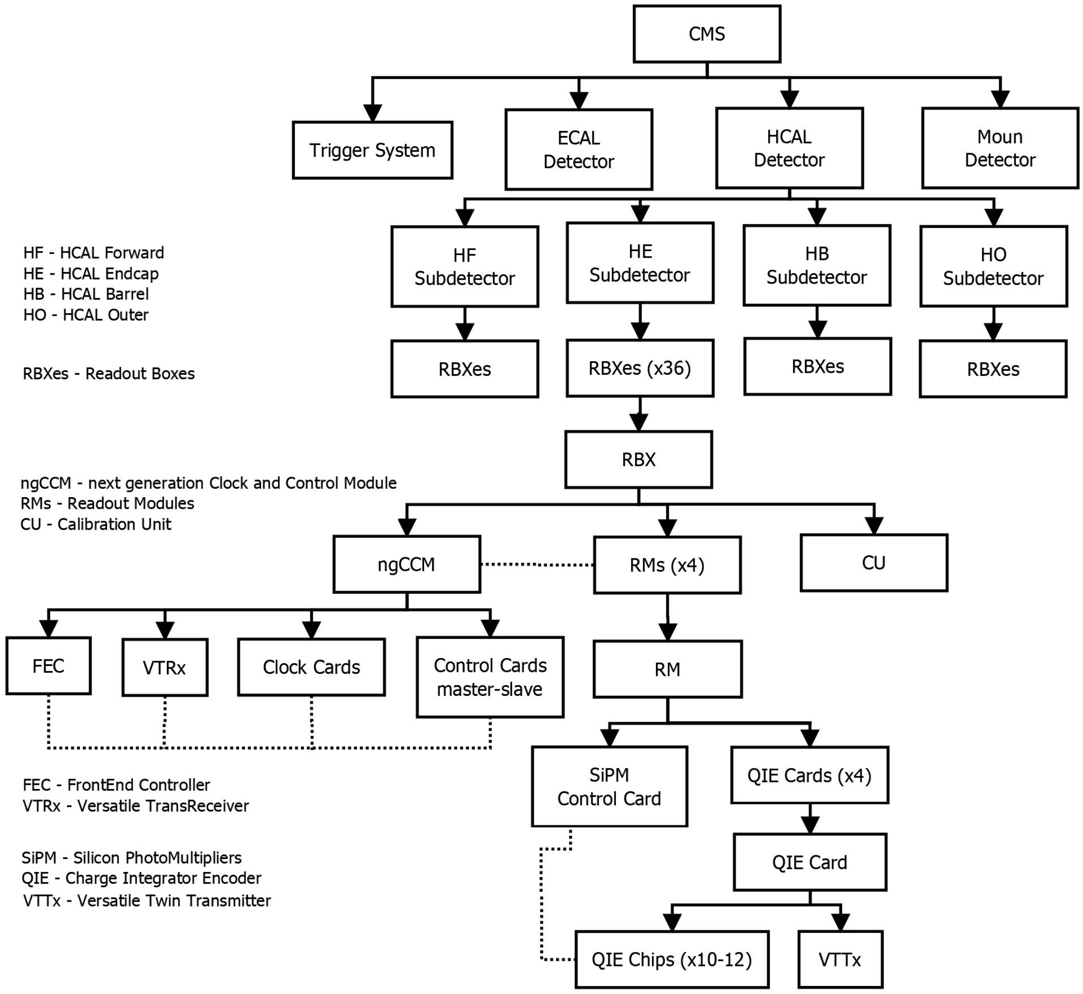


Figure 1.3: Hierarchical system diagram of the RBX system of the HCAL.

within 20 minutes before the beams converge and collide at one of the LHC detector sites. The LHC consists of three major systems to accelerate and collide beams: 1) radiofrequency metallic chambers to resonate at specific frequencies that allow radio waves to transfer energy and push forward the passing particle bunches, 2) various types of magnets to focus the beam particles closer together and keep the beams in the circular path, and 3) particle detectors are placed around the collision point to record the particles that emerge from the collisions.

The LHC consists of several experiments on its sites, and its ring holds several detectors for these experiments. The four major detectors of the LHC are *A Toroidal LHC Apparatus* (ATLAS), *Compact Muon Solenoid* (CMS), *LHC beauty* (LHCb), and *A Large Ion Collider Experiment* (ALICE) (see Fig. 1.4). Each detector studies particle collisions from a different perspective with different technologies. ATLAS at *point 1* (P1) and CMS at *point 5* (P5) are the two high-luminosity general-purpose detectors at the LHC, located in diametrically opposite sections; the detectors employ different technical approaches but aim to support each other's findings.

The CMS experiment is a general-purpose detector for high-energy physics at the LHC and located at P5 in the village of Cessy, in France [3,59]. The experiment aims to investigate collisions at the TeV scale, study the properties of the Higgs boson,

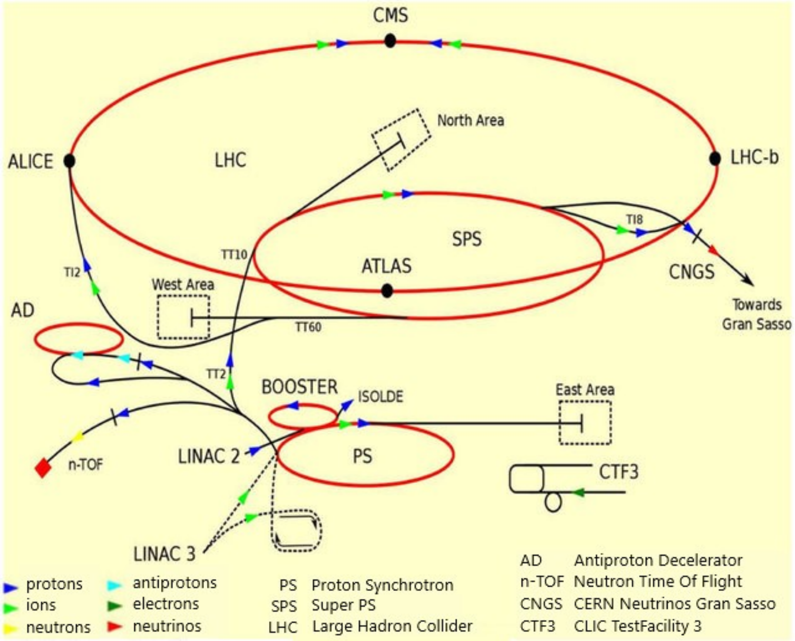


Figure 1.4: LHC particle injection, acceleration and collision chain [1]. Proton-proton collision: LINAC 2 → PS → SPS → LHC. Heavy-ion collision: LINAC 3 → LEIR → SPS → LHC. The PS, SPS, and LHC are 628 m, 7 km, and 27 km in circumference, respectively.

and search for evidence of new physics beyond the SM. Particle data collected with the CMS detector are utilized in many aspects of modern particle physics studies, notably the discovery [60] and characterization [61] of the Higgs boson. The CMS detector consists of two calorimeters—the *electromagnetic* (ECAL) and the *hadronic* (HCAL) to detect electrons, photons, and hadrons—and several *muon* detectors (see Fig. 1.5) [59]. The calorimeters measure the energy deposition of incident particles by causing them to interact and lose their energy in the calorimeter materials (see Fig. 1.6). The CMS also contains a Silicon tracker for particle momentum estimation by measuring the spatial trajectory of the particles. The energy and momentum profiles allow particle type identification.

The calorimeters of the CMS detector are highly segmented to improve the accuracy of energy deposition profile measurement and particle identification. The segmentation geometry of the detector is represented using η and ϕ spaces—also known as *pseudo-rapidity* and *azimuth*, respectively (see Fig. 1.7a). The z-axis lies along the incident beam direction and the ϕ is azimuthal angle between the x and y axis, while η is calculated from the polar angle θ_{cm} between z and $xy-$plane as:

$$\eta = -\ln(\tan(\theta_{cm}/2)) \tag{1.1}$$

where the x, y, and z are orthogonal axis of the the cylinder, the θ_{cm} is the center-of-mass scattering angle, and the ln is a natural log function. The $\eta - \phi$ space corresponds to a rectangular coordinate system representing an outgoing particle's direction from the center of the detector (where the collision occurs) (see Fig. 1.7). Particles traveling in the same direction lie near each other in $\eta - \phi$ space.

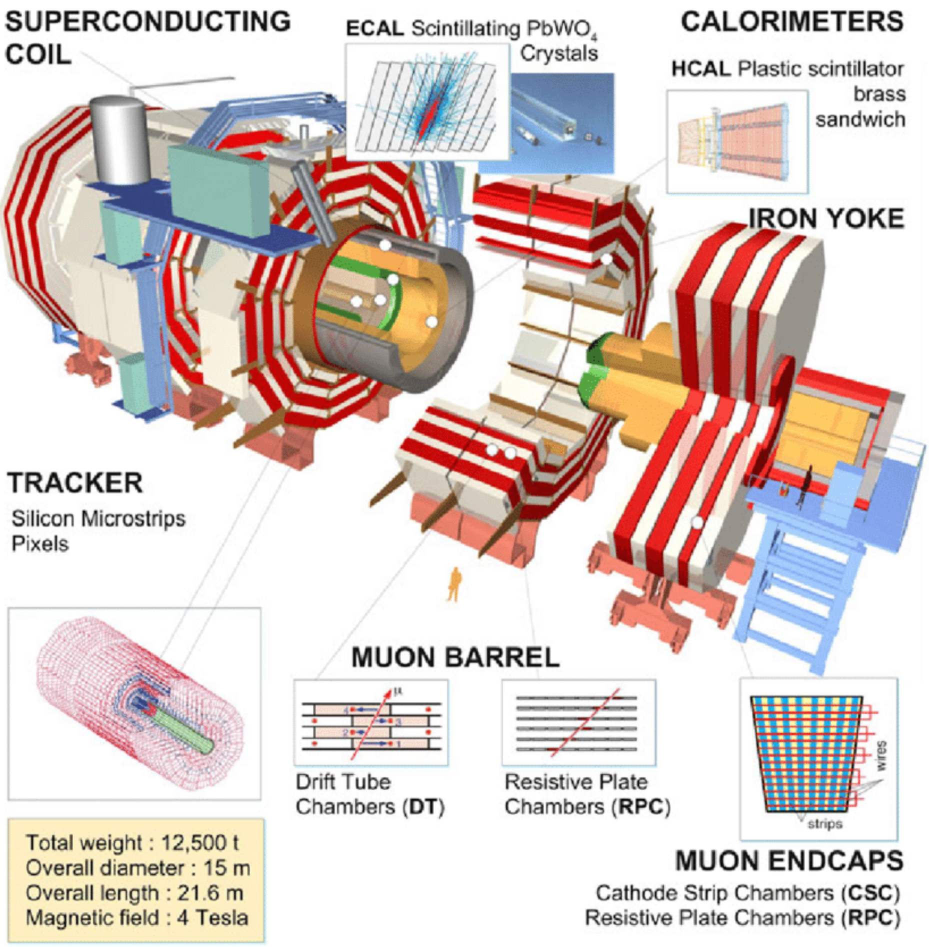


Figure 1.5: Overview of the CMS detector [2,3]. The detector is built around a huge solenoid magnet that generates a magnetic field of 4 Tesla—about 100K times that of the Earth.

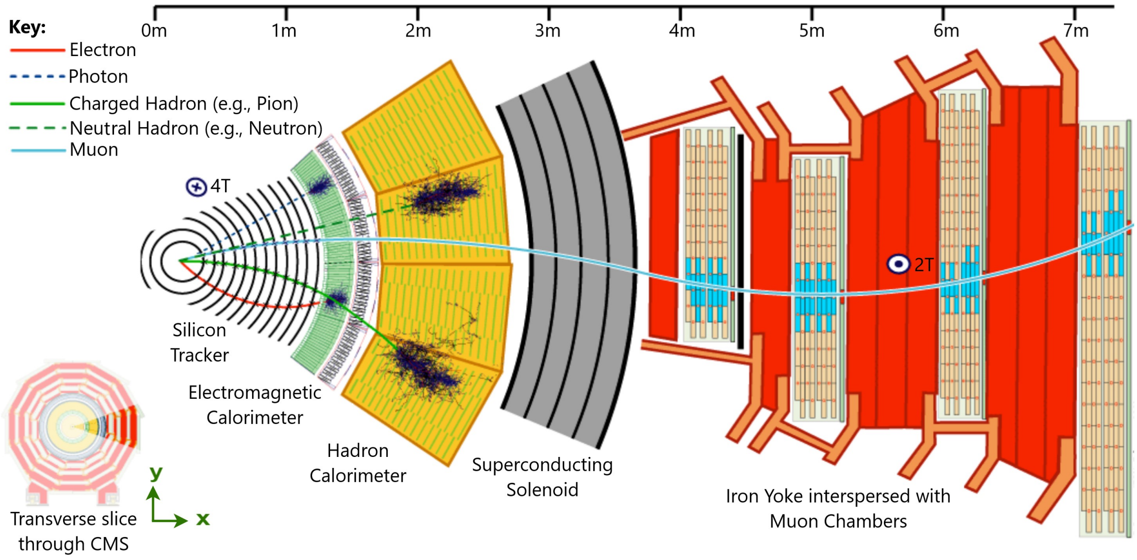

Key:
Electron
Photon
Charged Hadron (e.g., Pion)
Neutral Hadron (e.g., Neutron)
Muon
0m
1m
2m
3m
4m
5m
6m
7m
4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron Yoke interspersed with Muon Chambers
Transverse slice through CMS
y
x

Figure 1.6: A transverse slice through one segment of the CMS detector indicating the responses of the various detecting systems to different types of particles [4].

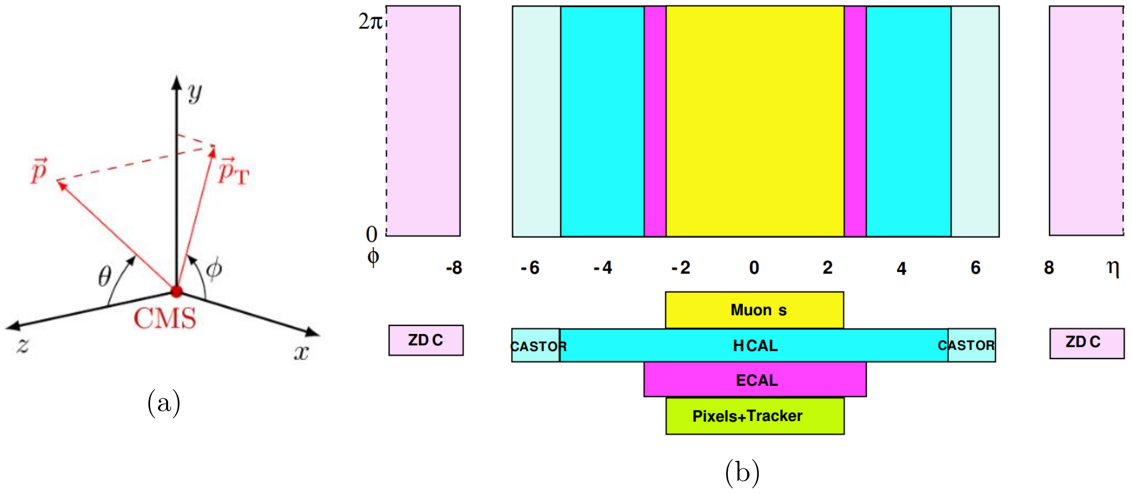


Figure 1.7: The pseudo-rapidity (η) and azimuth (ϕ) segmentation of CMS: a) geometry axes and angles of the CMS with respect to collision intersection point [5], and b) range of η and ϕ values for the tracker, calorimeters, and muon detectors [4]. The CASTOR and ZDC detectors are small dedicated Cerenkov calorimeters with hadronic and electromagnetic sections and a few hundred readout channels.

1.2.1 The Data Acquisition System

The LHC provides proton-proton and heavy-ion collisions at high interaction rates. It embraces a sophisticated *data acquisition system* (DAQ) to effectively process, store, and distribute the immense volume of collision data. The beam crossing interval at the LHC is 25 ns for proton beams—corresponding to a crossing frequency of 40 MHz. A drastic rate reduction has to be achieved since storing and processing the large amount of data associated with the resulting high number of events is impractical. The CMS employs a trigger system to drastically reduce the rate by $O(10^6)$ [59]. The trigger system performs the physics event selection process in two steps: 1) the hardware-based *level-1 trigger* (L1), and 2) the software-based *high-level trigger* (HLT). The L1 trigger reduces the incoming average data rate to a maximum of 100 kHz by processing fast trigger information on coarsely segmented data from the calorimeters and the muon system while holding events with interesting signatures in pipelined memories in the frontend electronics. The HLT is a faster and streamlined version of the CMS offline reconstruction software performing complex calculations to filter physics events on a multi-processor farm and further reducing the rate by $O(10^3)$. It consists of several evolving algorithms to meet the trade-offs between the complexity of the execution on the available computing power sustaining output rate and the particle event selection efficiency [62].

The collision data of the CMS experiment is stored with unique run identification (known as *RunId*)—each run contains thousands of luminosity sections (also called lumisections). A *lumisection* (LS) corresponds to approximately 23 seconds of data taking and comprises hundreds or thousands of collision events—particle hits. The LHC uses "Tiered" computing infrastructure to handle further analysis and event reconstruction after the HLT. The *worldwide LHC computing grid* (WLCG) unites high-energy physics institutions worldwide in distributed computing infrastructure. The WLCG aims to develop tools to make access to resources transparent and achieve efficient distributed computing services. The WLCG combines about 1.4 million computer cores and 1.5 exabytes of storage from over 170 sites in 42 countries distributed in three tiers: 20% at Tier-0 (CERN), 40% at Tier-1 (13 big scientific and academic institutions), and 40% at Tier-2 (150 institutions). The data stream is split into *primary datasets* (PD) based on the HLT results to group events with related topology in the same PD to ease consumption and limit replication of events.

1.2.2 The Hadron Calorimeter

The Hadron Calorimeter (HCAL) of the CMS detector is responsible for measuring the energy of hadronic showers originating from the LHC collisions (see Fig. 1.6). The HCAL also assists in indirectly measuring uncharged particles such as neutrinos. The HCAL has four major subdetectors covering different segments in the CMS detector: the *HCAL Barrel* (HB), *HCAL Endcap* (HE), *HCAL Outer* (HO), and *HCAL Forward* (HF) [3, 6, 63–66] (see Fig. 1.8). The HB and HE are sampling calorimeters with a brass absorber and active plastic scintillators to measure the energy depositions [64, 65]. The HO is a tail-catcher for hadronic showers that is useful for muon identification [65], and the HF is a Cherenkov calorimeter with a steel absorber and quartz fibers that collect Cherenkov light [66]. The central HB and HE subdetectors surround the ECAL and are fully immersed within the strong magnetic field of the solenoid, where the HB are joined hermetically with the barrel extending out to $|\eta| = 1.4$ and the HE covering the overlapping range $1.3 < |\eta| < 3.0$. The HF is located 11.2 meters from the interaction point and extends the pseudo-rapidity coverage—overlapped with the HE—from $|\eta| = 2.9$ to $|\eta| = 5$. The central shower containment for $|\eta| < 1.26$ is improved with the HO—an array of scintillators located outside the magnet.

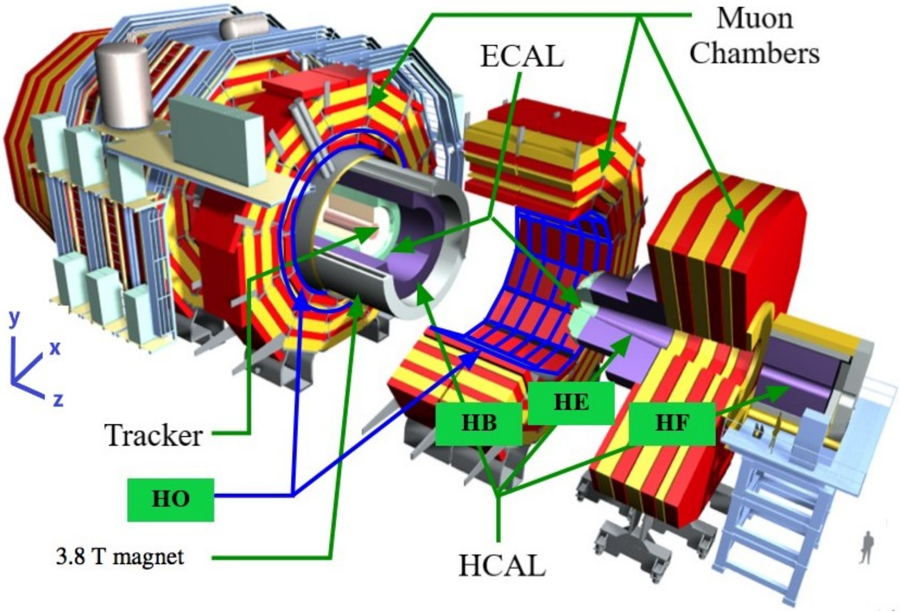


Figure 1.8: The HCAL subdetectors in CMS: the HB, HE, HO, and HF [3].

1.2.3 The HCAL Front-End Electronics

The HCAL electronics systems are composed of two sections, known as the *front-end electronics* (FE) and the *back-end electronics* (BE) (see Fig. 1.9) [6]. The FE consists of the HCAL components that are responsible for particle reading and digitization, and the BE receives optical data streams through 4.8 Gbps fiber links from the HE and is composed of data preprocessing systems. The FE is contained in compact modules that plug into a backplane and mechanical assembly firmly mounted on the detector. The FE electronics are divided into sectors of *readout boxes*—hereafter called RBXes—that house the FE electronics components and provide voltage, backplane communications, and cooling. The FE of the HCAL subdetectors have a common purpose and share similar designs and technologies but also include differences. Keeping the designs similar allows resource sharing, reduces the overall engineering needed, and simplifies long-term maintenance. The HCAL uses a continuous readout FE—no memory in the pipeline—transmitting the reading for every bunch-crossing to the back-end electronics. This strategy allows the data processing to be adjusted in the firmware of the BE and new features added without challenging and risky modifications to the FE. The HCAL BE connects to the CMS trigger, DAQ, and DCS systems. It adopts the *μTCA* architecture from the telecommunications standard, and the structure of each crate typically includes twelve *μTCA HCAL trigger/readout* (uHTR) cards and an *advanced mezzanine card* (AMC13). The uHTR cards are responsible for receiving the front-end data links, calculating and transmitting event trigger primitives, and holding the L1 readout pipeline—retaining the data samples waiting for the L1 decision. The AMC13 card in each crate is responsible for data acquisition and distribution of the LHC clock and fast control signals.

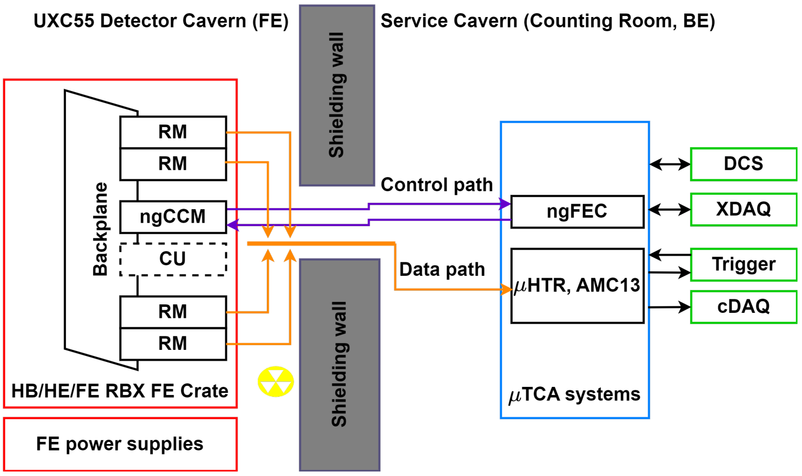


Figure 1.9: Crate layout structure diagram of HCAL front-end (FE) and back-end (BE) electronics (plot inspired by [6]).

The FE follows a common design across the HB, HE, and HF calorimeters [6]. Each HCAL subdetector is composed of RBXes that house several electronic components that are common across the designs. We will describe the RBX system components and their operations since we focus on monitoring the RBX of the HE and HB subdetectors. The HB and HE subdetectors (HBHE) are arranged into the minus and plus hemispheres—HBP and HEP, and HBM and HEM, respectively. Each half is further divided into eighteen identical wedges, and one RBX receives light signals from each wedge. Fig. 1.10 illustrates the RBX numbering and geometry of the HE. The RBX represents the smallest unit of front-end control and power—with each RBX consisting of a water-cooled aluminum shell housing the FE particle data acquisition, control, and communication electronics [6, 63, 64, 66]. The RBX electronics components consist of low voltage distribution, high voltage distribution, digitization *readout modules* (RMs), control and synchronization *next-generation clock and control module* (ngCCM), and a *calibration unit* (CU) (see Fig. 1.11). Fig. 1.3 illustrates the hierarchical decomposition structure of the electronics components of the FE of the HE.

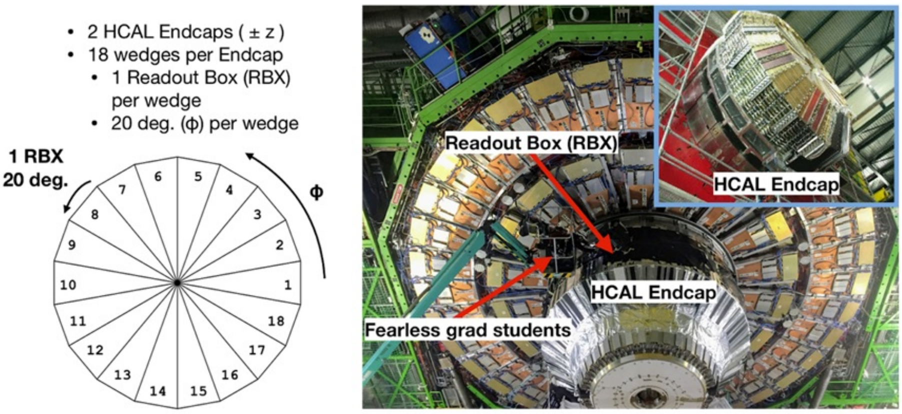
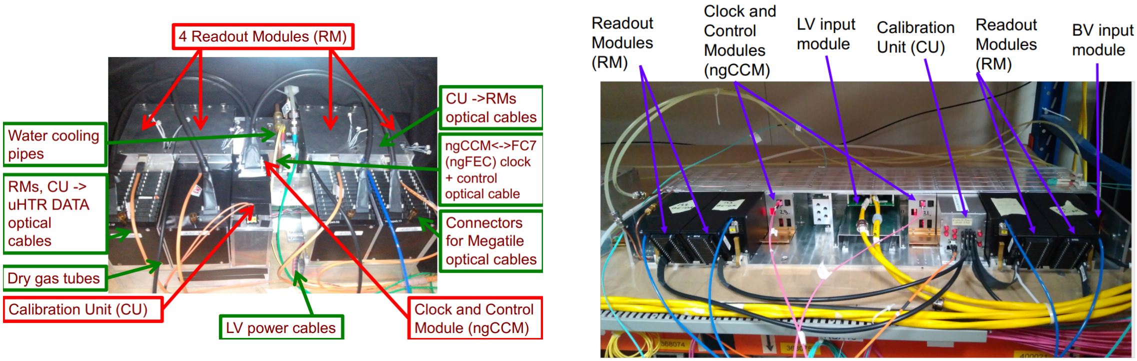


Figure 1.10: The HE on CMS detector: (left) arrangement of eighteen RBXes, and (right) installation position of the HE on the CMS.



(a)

(b)

Figure 1.11: Readout box of the a) HE and b) HB [7]. The RBX consists of four RMs, a ngCCM, a CU, and power supply and communication modules.

The RMs receive analog light pulses from the calorimeter, digitize them in 25 ns time-slices using *charge integrating and encoding* (QIE) electronics, and transmit the resulting digital data through optical fibers. The ngCCM provides backend-to-frontend communication, control, and clock distribution for the FE of the detector.

Failures in the ngCCM induce communication loss to the RBX—affecting a significant number of detector channels, and such occurrences would be a severe fault. The ngCCM is thus designed to be robust against failures on the parts of the ngCCM card and the optical fiber links. The ngCCM of the HBHE has two control cards—designated as J14 and J15—that work in a master-slave fashion. Each frontend electronics crate will contain one CU, and its purpose is to simulate analog light signals using an LED pulser—without actual particle collision—during the detector operation testing phases.

The HBHE front-end particle detection system is built on brass and plastic scintillators, and the produced photon is transmitted through the wavelength-shifting fibers to *Silicon photomultipliers* (SiPMs) (see Fig. 1.12). Each RBX houses four RMs for signal digitization [8]. Each RM has 64 (48) SiPMs—64 for HB and 48 for HE—and each SiPM is connected to its corresponding QIE chip (ASIC). A QIE integrates charge from one SiPM at 40 MHz, and four *field programmable gate array* (Microsemi Igloo2 FPGA) based *QIE cards* serialize and encode the data from 16 (12) QIE chips (see Fig. 1.12). The encoded data is optically transmitted to the back-end system through the CERN *versatile gigabit transceiver* (GBT) at 4.8 Gbps. The GBT is packaged either as a *transceiver* (VTRx) or a *dual-twin-transmitter* (VTTx) in an enhanced *small form-factor pluggable* (SFP$^+$). Seventeen detector scintillator layers are read out in four and seven groups—referred to as *depths*—for the HB and HE, respectively. The light from scintillators in any given group is optically added together by sending it to a single SiPM. More channels allow for a more refined depth segmentation, which is ideal for precisely calibrating the depth-dependent radiation damage on the HCAL. The performance for physics quantities is recovered with the increased light yield and better calibration [49].

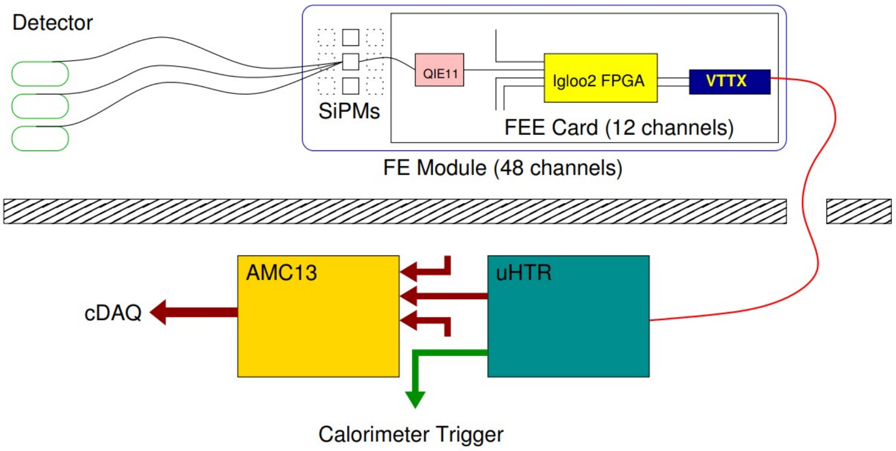


Figure 1.12: The data acquisition chain of the HE—including the SiPMs, the FE readout QIE card, and the optical link to the BE electronics [8]. The QIE card contains twelve QIE11 chips for charge integration, one Igloo2 FPGA for data serialization and encoding, and one VTTx optical transmitter.

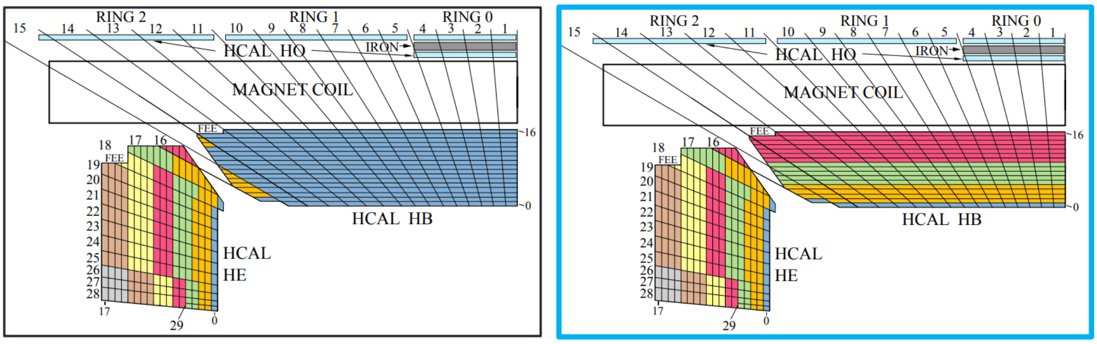
1.2.4 Major Upgrades on the HCAL Front-end Electronics

The LHC has undergone a series of two long shutdowns—designated as LS1 (2013-2014), and LS2 (2018-2022)—and three operation runs—named Run-1 (2009-2013), Run-2 (2015-2018) and Run-3 (2022-). The LHC was maintained and upgraded during the LS2 to implement the high luminosity LHC (HL-LHC) project, which aims to increase luminosity by a factor of ten [6]. The CMS upgrade program during LS2 took advantage of new technologies and implemented three major upgrades: a new pixel detector with high data rates, an improved L1 with higher granularity and additional processing capabilities, and an upgraded photo-detector and electronics to reduce background signals and improve measurement of jets and missing-energy at high pileup [6–8].

The HB, HE, and HO calorimeters were all originally fitted with *hybrid photodiode* (HPD) transducers—chosen for their magnetic field tolerance and gain of $> 2{\times}10^3$; each HPD is segmented—providing channels of optical-to-electrical conversion. Several weaknesses were identified in the HPDs during operation—mostly due to the high electric field required (8 kV across a 3 mm gap) [6]. The most significant of these was the appearance of electrical discharges in the device when high voltage was applied. The effect was particularly severe for the HO calorimeter—required substantial reductions in voltage from 8 kV to 6.5 kV in much of the detector. The discharge effect was a source of high-amplitude noise and response gain drift that risked the reliability and longevity of the phototransducers. The CMS research and development program has invested significant effort to replace the HPDs with SiPMs, bringing better and more stable performance to the HB and HE detectors [7, 8]. The SiPM—a multipixel Geiger-mode avalanche photodiode (APD) device—is an efficient, relatively compact, and low-cost alternative; it provides gains between 10^4 and 10^6 using voltage supply of less than 100 V, and photon detection efficiencies in the range of 20% to 40%. The high performance of the SiPM devices—coupled with recent developments in data link technology—significantly increases the segmentation in the HB and HE calorimeters (see Fig. 1.13). A finer segmentation allows better tracking of hadronic shower development—important for particle-flow analysis [6, 7]. It also enables better management of the radiation damage that occurs in the high η region of the HE calorimeter, reducing the response of the individual tiles. We have studied machine learning for the HB and HE particle acquisition monitoring using online DQM histogram map data from 2018 (Run-2) and 2022 (Run-3). The HPD upgrade to the SiPM was accomplished for the HE in 2018 and the HB in 2019.

1.3 The CMS Experiment Monitoring Systems

The CMS experiment dedicates significant effort to continuously monitor its particle data quality and infrastructure. The *detector control and safety systems* (DCS/DSS) provide real-time and retrospective monitoring of the detector infrastructure. The *data quality monitoring* (DQM) detects and diagnoses particle data



(a) 2018

(b) 2019

Figure 1.13: Longitudinal segmentation (per given *phi*) of the HBHE subdetectors [7]. The numbers and colors denote $i\eta$ and depth segment, respectively. $i\eta$ is an integer denotation of the η axis. The HB cover the inner $|i\eta| \leq 16$ followed by the HE on $16 \leq |i\eta| \leq 29$. The HE was upgraded to seven depths in 2018, while the HB had two depths in 2018 and was upgraded to four in 2019.

quality issues [39]. There are limited works on infrastructure monitoring, whereas promising studies have been proposed leveraging the DQM automation with ML algorithms [39, 40, 50, 67]. We will briefly describe the DCS/DSS and DQM systems in the following subsections.

1.3.1 The Detector Control Systems

The CMS experiment employs the detector control and safety systems (DCS/DSS) to ensure safety protection and proper operation during experiments, respectively [57, 59]. The DCS ensures the proper functioning of the CMS experiment—monitoring the detector's subsystems, electronics, and surrounding environment—to capture high-quality data. It provides bookkeeping of detector parameters and safety-related functions—including alarm handling and controlling critical components through software access [59]. The DCS communicates with external entities like the run control of the DAQ and serves as the interface between CMS and the LHC. It uses SCADA software to model detectors and support services as finite state machine nodes arranged in a tree-like hierarchy, representing the logical structure of the detector, where commands flow down, and states and alarms are propagated up. Both DCS and DSS require high availability and reliability [57].

The DCS actively monitors CMS throughout active collision data-taking and manages several monitoring and control tasks, such as high voltage (HV) and low voltage (LV), gas system and environmental parameters, and communication with external systems [57]. The monitored quantities include but are not limited to voltage levels at specific points in the system, electrical currents, temperature and humidity of components, optical light levels, error counts for data transmission between modules, etc. The DCS provides a real-time status log of several CMS components, such as detectors, trigger systems, communications modules, and power supplies. It is an essential tool for shift crew members, detector subsystem experts, operations coordinators, and those performing physics analyses.

The potential limitation of the DCS is that only a relatively small set of sensors are monitored in realtime to alert physicists about rather critical problems. Most of the logged quantities are currently accessed reactively for debugging purposes when significant issues are encountered in the detector components. The proactive monitoring relies on threshold-based alerts on quantities like temperature or bias voltage, while the retrospective debugging involves visually inspecting a large and diverse set of diagnostic sensor signals [22]. The shortcomings of this approach are 1) only a small subset of quantities is monitored in realtime, 2) the decision preset ranges are static, and 3) potentially useful correlations among quantities are not considered. The approach has scalability issues—human resources—to analyze large quantities, and the monitoring is naturally subject to human arbitration. Predicting faults of the detector FE electronics is essential to maintain the acquisition system capability and yield data with greater quality [68]. Active monitoring of detector system anomalies on more comprehensive variables extends the monitoring range to catch previously missed faults [22]. Machine learning models are thus being explored for system monitoring through AD automation on sensor data at the LHC [41, 52, 56, 69, 70]. AD models enable the automatic monitoring of large sets of variables and have the potential to capture detector performance issues that escape the DCS monitoring.

1.3.2 The Data Quality Monitoring

The detector and collision data offline processing complexity requires continuous data quality monitoring. Shifters and physicists at CMS monitor the collision quality and select data usable for analysis; they look for unexpected issues that could affect the data quality—e.g., noise spikes, dead areas of the detector, and calibration problems [9]. The data quality monitoring (DQM) system of CMS provides online and offline monitoring [49] (see Fig. 1.14): 1) the *online DQM* is real-time monitoring during data acquisition streamed from the HLT, and 2) the *offline DQM* provides the final fine-grained data certification 48 hours after the collisions were recorded. The online DQM identifies emerging problems using reference distributions and predefined tests to determine known failure modes [71, 72]. It populates a set of histogram maps on a selection of events that are sensitive to detector functioning and operations for real-time data quality monitoring. The experts inspect summary plots and alarms—produced by expert-defined algorithms and rules—of the online DQM to spot and diagnose problems. The list of runs and lumisections where all sub-detectors are in a good state are stored in a "*Golden JSON*" file.

ML leverages the DQM to address particular challenges, such as:

- *Latency*: Human intervention or/and thresholds require sufficient statistics.
- *Volume budget*: The amount of data a human can process in a finite time is limited.

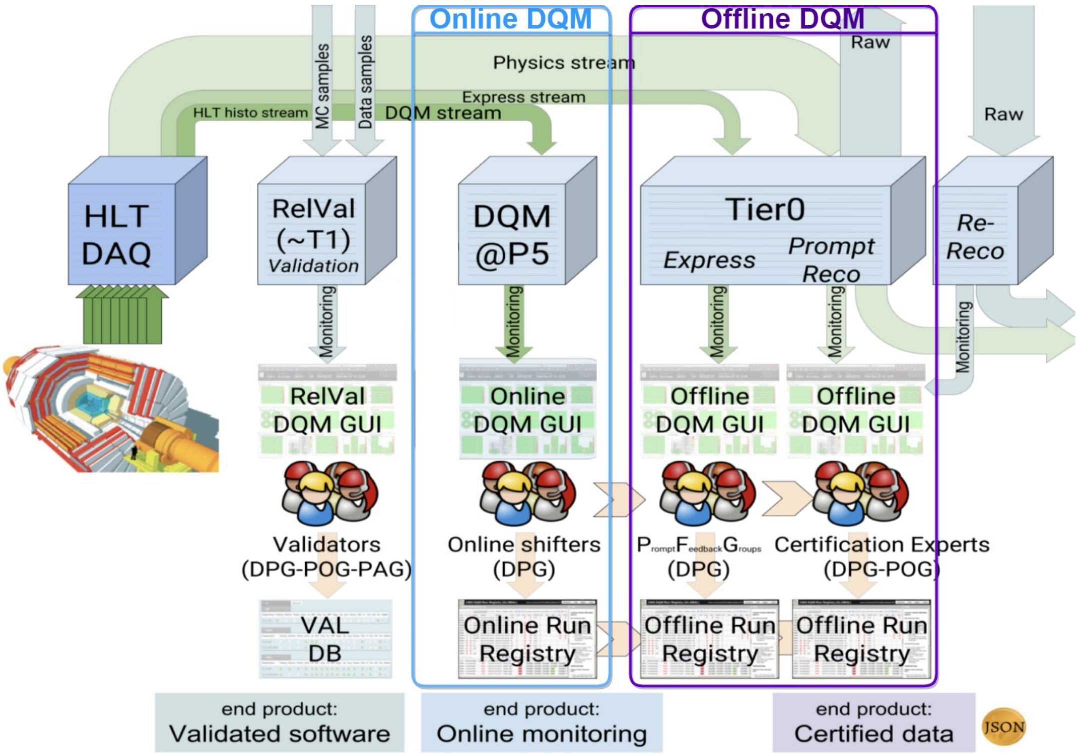


Figure 1.14: The DQM system of the CMS experiment [9]. The run registries hold the configuration and DQM status of the collision runs. The DQM generates "Golden JSON" that records the validated good runs for further physics analysis.

- *Scaling*: Rule-based approaches do not scale and assume limited potential failure scenarios.
- *Dynamic running conditions*: Reference samples change over time.
- *Human resources*: Training shifters and maintaining instructions is expensive.

Developing ML models using the DQM quantities comes with some constraints—including data normalization to handle variation in experimental settings, the granularity of the failures to spot, and limited availability of the ground truth labels [40]. Machine learning algorithms have been considered for DQM AD applications across several CMS subsystems [39, 40, 49, 50]; but, only a few employ end-to-end DL on spatial DQM data, and temporal models are relatively unexplored [39, 40].

1.4 Research Questions and Proposed Solutions

Physicists and shifters continuously monitor the data-taking process of the current complex LHC detectors through hundreds of expert-defined histograms. These histograms alert them to unexpected deviations from a reference. New types of problems sometimes go unnoticed for a significant period of time because the experts did not foresee it [22]. We will present below the research gaps in detector monitoring and outline our research questions along with the highlights of our proposed solutions.

1.4.1 Infrastructure Monitoring

The ongoing complexity of the LHC has led to a need for greater automation. The quality of the collected data depends on the detectors' operational well-being during particle data-taking. Rapid identification and resolution of detector system anomalies will result in a better quality of high-precision particle data. Several issues—including detector malfunctions and software problems—can affect the collision particle quality acceptance rate. The sensors of detector systems—e.g., voltage, current, temperature, humidity, etc.—are continuously monitored to ensure the quality of the recorded data in CMS. The CMS experiment partly relies on retrospective monitoring and debugging approaches—involving the visual inspection of a large and diverse set of diagnostic sensor signals of the electronic components of the detectors. Data-driven AD on the monitoring variables can be employed to trigger flags related to the detectors' operational status. There has been thus far limited effort to exploit time series sensor data sets despite the acknowledged potential for future CERN automation technology challenges [39]. Refs. [52, 56] have explored deep learning on time series data for safety monitoring of the magnetic systems of the LHC accelerator. There have been limited tangible efforts in exploring ML to support system monitoring automation of the HCAL through anomaly detection models [69].

We have identified the following three research questions on machine learning for diagnostics sensor monitoring of the HCAL given the aforementioned challenges:

- RQ1: How can machine learning enhance the automated identification of anomalous behavior in the HCAL detector systems?
- RQ2: How can machine learning techniques enable the prediction of anomalous behavior in the HCAL detector systems?
- RQ3: How can anomaly causal discovery improve system understanding and aid root-cause identification of problems?

We highlight below our solutions addressing the above research questions:

- We will present ML for AD that automates and enhances the identification of HCAL anomalies from diagnostics monitoring sensor quantities to answer RQ1 (**Paper-1** in Ref. [41]). We propose CGVAE—a data-driven unsupervised anomaly detection using a deep learning model—for the HCAL Encap detector ngCCM system monitoring from multivariate time series sensor data. The CGVAE model comprises a variational autoencoder with convolutional and gated recurrent unit networks for fast localized feature extraction, long temporal characteristics capturing, and descriptive representation learning. The CGVAE employs encoded latent feature- and reconstruction-based metrics for anomaly detection to mitigate signal reconstruction overfitting on anomalous patterns. The model integrates feature attribution algorithms to explain the contribution of the input sensors to the detected anomalies. The experimental evaluation on large sensor data sets of the HCAL demonstrates the efficacy of the proposed model in capturing a wide range of anomalies—including slow drifts.

- We address RQ2 through an end-to-end deep learning approach for anomaly prediction using early warning symptoms on multivariate time series sensor data (**Paper-2** in Ref. [70]). We will introduce AnoP—a long multi-timestep anomaly prediction system based on unsupervised attention-based causal residual networks—to raise alerts for anomaly prevention. The experimental evaluation of AnoP on the HCAL RBX sensors shows an encouraging efficacy. The AnoP predicts anomalies up to seven days ahead. We have also discovered previously unknown anomalies in the HCAL sensors.

- We will present two solutions to address RQ3: 1) lightweight mechanism for system and sensor interconnection discovery, and 2) computationally efficient graph causality model discovery approach on sparse time series anomaly data.

  We propose a lightweight mechanism for discovering the interconnection of sensors that enables fast detection of divergent behaviors in large sets of multivariate environments (**Paper-6** in Ref. [73]). The method generates clusters and association links—allowing us to identify abnormal system states and the cause of abnormality—faulty sensors. We apply the exploration approach using a similarity distance matrix to detect diverging sensors and systems behavior on the multivariate monitoring sensor data of the RM from thirty-six RBXes systems of the HE detector. The results demonstrate the clustering of systems and sensors consistent with the actual expected configuration of the detector and systems with unusual sensor readings form divergent clusters.

  The causal discovery can lead to quicker and more effective fault diagnostics once an anomaly is detected in the realm of system monitoring (**Paper-7** in Ref. [74]). Identifying anomaly causes on large systems involves investigating a more extensive set of monitoring variables across multi-subsystems. Causal graphs provide an intuitive presentation, but the computational cost of graph learning quickly increases for large and high-dimensional data sets. We propose AnomalyCD—a computationally efficient approach for discovering causality—exploiting the unique attributes of large-scale anomaly flag data sets. The framework consists of multiple strategies to overcome the challenges of generating causality graphs through unsupervised online AD, sparse data and link handling, anomaly data behavior aware custom independence test, and causal graph learning using the PCMCI algorithm. The experiment on the RM of the HE-RBX demonstrates that the approach accurately detects outliers and generates causal networks consistent with the actual physical circuit connections and environmental associations. The proposed framework substantially reduces the computational cost of causal graph discovery as evaluated on real and synthetic data sets.

1.4.2 Data Quality Monitoring

A small team at CMS has been investigating and developing ML-based automation for DQM (ML4DQM) since 2016 [39, 40, 50, 67]. The synergy in ML model development has thus far achieved promising results on spatial 2D histogram maps of the DQM for particular types of anomalies of the ECAL [39] and the muon detectors [40]. ML automation for the DQM of the HCAL has not been adequately explored. Monitoring the particle data acquisition sensors of the HCAL poses multidimensional challenges—3D histogram maps—due to its depth-wise segmentation. Previous studies have only considered extreme anomalies—no reading dead and high noisy reading hot—of the particle sensing channels. Detecting degrading channels is relevant for quality deterioration monitoring and early intervention, but they are often challenging to detect; for instance, the improperly tuned bias voltage on the HCAL physics particle sensing channels caused non-uniformity in the particle hits map of the DQM, but the channels were neither dead nor hot [75]. The previous efforts have not been focused on temporal models despite the acknowledged potential in exploiting temporal context for enhancing the AD efficacy [39,52]. Capturing subtle anomalies—slow system degradation—makes temporal models appealing in raising early alerts before ultimate system failure. Developing ML monitoring tools for the DQM with dynamic behavior is challenging due to the difficulty of data renormalization and the lack of adequate annotated ground truth with different operating conditions—the LHC beam and intensity configurations.

We have specified the following two research questions on machine learning for online DQM of the HCAL:

- RQ4: How can temporal attributes be exploited to leverage the automated ML-based AD of the DQM with enhanced detection of anomalies manifested at the lumisection granularity of the HCAL?

- RQ5: How can employing transfer learning for TSAD mitigate the challenge of limited annotated data for varying operating conditions in the HCAL?

We highlight below our solutions addressing the above research questions:

- We address RQ4 by developing a semi-supervised spatio-temporal (ST) AD monitoring for the physics sensing channels of the HCAL using the three-dimensional particle hit maps of the DQM (**Paper-3** in Ref. [76] and **Paper-4** in Ref. [77]). We present GraphSTAD, which employs convolutional, graph, and recurrent neural networks to detect anomalies by capturing ST characteristics induced by particles traversing the calorimeter and shared backend circuitry. We validate the accuracy of the proposed AD system in capturing abnormal channels in the LHC collision data. The system achieves production-level accuracy and is integrated into the CMS core production system—for real-time detection of several anomaly types.

- We address RQ5 using transfer learning (TL) to accelerate model training and enhance accuracy for the DQM AD models in varying environments (**Paper-5** in Ref. [78]). The deployment of AD models in new environments is often hindered by the limited amount of cleaned data [79] despite the promising accomplishment of unsupervised approaches in monitoring applications [80–82]. We present both the potential benefits and limitations of TL within the context of AD in CMS. We have transferred models trained on data collected from one HCAL Endcap to the HCAL Barrel subdetector. We discuss different TL configurations on the GraphSTAD model transferred from the HE to the HB subdetector. Our results demonstrate that TL can effectively extract and reconstruct ST features on a new target dataset. The TL achieves promising AD accuracy while substantially reducing the number of trainable parameters. The TL also improves robustness against anomaly contamination in the training dataset.

1.5 Methodology Overview

This study focuses on developing data-driven monitoring tools for the HCAL front-end electronics systems. The HCAL presents challenges for machine learning modeling: 1) large, heterogeneous, and high-dimensional data sets, 2) mostly unannotated data sets, and 3) diverse operation configurations. The modeling requires intensive data curation, preparation, and scalable modeling approaches. We follow general- and problem-specific methodologies to address the challenges. Our approach involves a divide-and-conquer method, contextual modeling on multivariate temporal data, and unsupervised end-to-end deep learning designs (see Fig. 1.15).

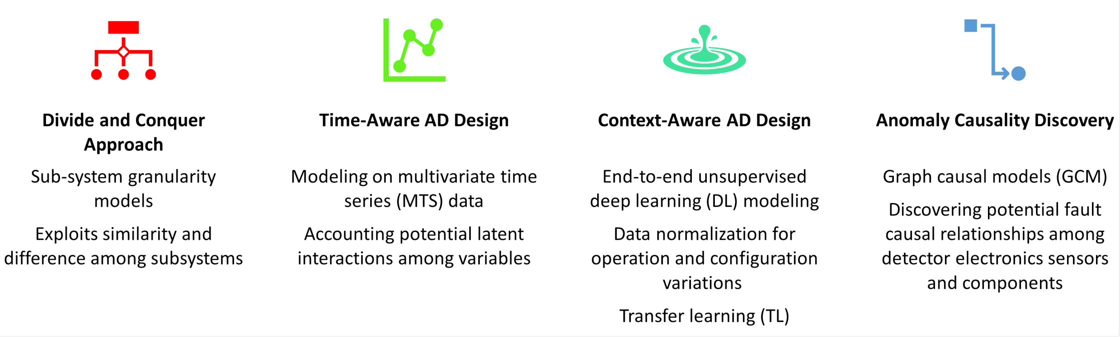

Divide and Conquer Approach
Sub-system granularity models
Exploits similarity and difference among subsystems
Time-Aware AD Design
Modeling on multivariate time series (MTS) data
Accounting potential latent interactions among variables
Context-Aware AD Design
End-to-end unsupervised deep learning (DL) modeling
Data normalization for operation and configuration variations
Transfer learning (TL)
Anomaly Causality Discovery
Graph causal models (GCM)
Discovering potential fault causal relationships among detector electronics sensors and components

Figure 1.15: Diagram of the general methodology.

We embarked on our study by identifying HCAL system monitoring challenges and proceeded to formulate the research questions. We have developed and deployed pertinent ML models for the CERN monitoring systems (see Fig. 1.16). We have modeled the complex systems with thousands of sensors through a divide-and-conquer approach in which the modeling is carried out at subsystem-level granularity. We have employed time context-aware end-to-end unsupervised deep learning models and transfer learning to account for variations in operation conditions and system configurations. Exploiting temporal context can significantly enhance AD performance by utilizing time-aware models; these models are ideal for early alert raising before total system failure by detecting subtle anomalies like gradual system degradation.

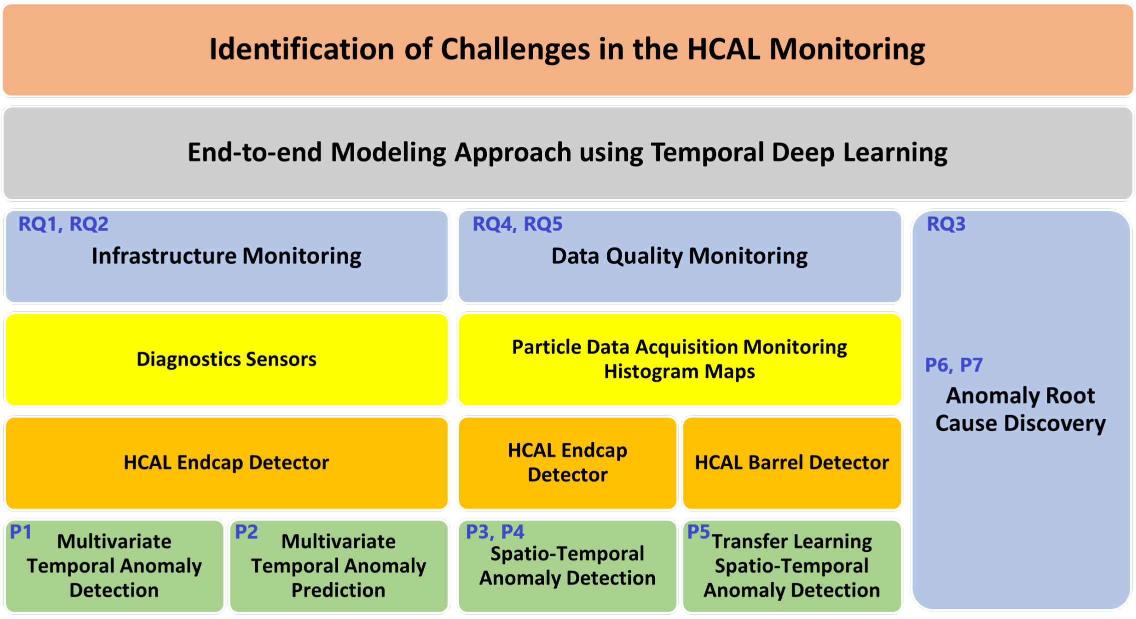

Identification of Challenges in the HCAL Monitoring
End-to-end Modeling Approach using Temporal Deep Learning
RQ1, RQ2
Infrastructure Monitoring
RQ4, RQ5
Data Quality Monitoring
RQ3
Diagnostics Sensors
Particle Data Acquisition Monitoring Histogram Maps
P6, P7
Anomaly Root Cause Discovery
HCAL Endcap Detector
HCAL Endcap Detector
HCAL Barrel Detector
P1
Multivariate Temporal Anomaly Detection
P2
Multivariate Temporal Anomaly Prediction
P3, P4
Spatio-Temporal Anomaly Detection
P5
Transfer Learning Spatio-Temporal Anomaly Detection

Figure 1.16: Our research workflow diagram.

1.5.1 Data Collection

Our study data sets were collected in 2018-2022 by the CMS experiment from the Run-2 and Run-3 collision operations of the LHC. The front-end electronics *diagnostics sensor data* and collision data quality monitoring *digi-occupancy map files* are retrieved from the *ngCCM server* and the DQM system of the CMS experiment, respectively. We familiarized ourselves with the monitoring variables by collaborating closely with CMS and HCAL experts, referencing HCAL documentation, and conducting extensive data exploration.

1.5.2 Machine Learning Models

We have primarily explored unsupervised deep autoencoder models for building contextual and robust multivariate time series anomaly monitoring systems. The modeling neural networks include *convolutional neural networks* (CNN) for robust feature extraction and capturing correlation among multivariate variables, *long short-term memory networks* (LSTM) and *gated recurrent unit networks* (GRU) for capturing long temporal dependency, *graph neural networks* (GNN) for their ability to learning non-Euclidean representation of data. We have utilized variational AE [83] to generate normally distributed probabilistic latent representation and regularize our autoencoder models. Multivariate reconstruction errors of the AD models and post-hoc ML output explanation methods, such as *SHapley Additive exPlanations* (SHAP) [84] and *Integrated Gradients* (IG) [85], are utilized for anomaly model output explanation. We have leveraged a constraint-based *Peter-Clark momentary conditional independence* (PCMCI) algorithm [20, 86] for causal discovery in time series anomaly data.

1.6 Thesis Roadmap

We organize the dissertation as follows:

- *Chapter-1: Introduction*: Discusses machine learning applications in the field of high-energy physics, describes the CMS experiment, presents our research questions and general methodology, and highlights our proposed solutions.
  - *Section-1: Machine Learning in High-Energy Physics:* Discusses machine learning in high-energy physics applications at LHC.
  - *Section-2: The CMS Detector System Overview:* Describes the use-case system of our study—the CMS-HCAL systems. It explains the LHC, CMS, and hardon calorimeter systems.
  - *Section-3: The CMS Experiment Monitoring Systems:* Describes the existing monitoring systems of the CMS experiment and their limitations.
  - *Section-4: Research Questions and Proposed Solutions:* Presents the research gaps, research questions, and highlights of our proposed solutions.
  - *Section-5: Methodology Overview:* Describes our general and problem-specific methods for addressing our research questions.
- *Chapter-2: Anomaly Detection and Prediction in Industrial Systems:* Reviews the related works on ML applications for fault monitoring applications—particularly in anomaly detection and prediction, transfer learning mechanisms, and anomaly causal discovery methods.
- *Chapter-3: Machine Learning for Time Series Modeling:* Discusses time series modeling and presents the architectures and the working principles of various popular deep learning models.
- *Chapter-4: Anomaly Detection:* Presents our study on (RQ1) deep learning models for infrastructure monitoring of the HCAL using diagnostics sensors of ngCCM and RM systems of the HE-RBXes, and (RQ4 and RQ5) online data quality monitoring through deep learning models on high dimensional DQM histogram maps.
- *Chapter-5: Anomaly Prediction:* Presents our study on (RQ2) long-horizon anomaly prediction on multivariate diagnostics sensors of the ngCCM system of the HE-RBXes.
- *Chapter-6: Anomaly Diagnostics:* Presents our study on (RQ3) multivariate multi-system interconnection exploration and scalable anomaly causal discovery on large sparse anomaly flags data streams of the HCAL.
- *Chapter-7: System Integration and Deployment:* Presents the deployment of our machine learning tools in the DESMOD dashboard and the CMS core production systems—the CMSSW.

- *Chapter-8: Conclusions:* Provides the summary of the scientific contribution of our study, discusses the research impact, and presents future research questions.

We will frequently use the mathematical notations presented in Table 1.1 throughout the thesis. We provide a generic explanation of the notations that remain valid unless explicitly stated.

Table 1.1: Mathematical notations used in the thesis.

Notation	**Remark**
$\mathcal{F}_{\beta}(\mathrm{X})$	Function—e.g., neural network model—with parameter β and input data X.
X	Multivariate matrix data with variable and time dimensions—$\mathrm{X} \in \mathbb{R}^{N \times T}$.
$\mathbf{x}$	Univariate vector data—$\mathbf{x} \in \mathcal{R}^{1 \times T}$ or $\mathbf{x} \in \mathbb{R}^{N \times 1}$.
x_t or $x(t)$	Univariate data reading at time t.
$\bar{\mathrm{X}}$	Predicated or reconstructed data of X.
$\mathbf{z}$	Latent representation feature data of X—$\mathbf{z} \in \mathcal{R}^{N_z \times 1}$.
$\mathbf{e}_i$	Prediction or reconstruction or error of the i^{th} variable.
$\mathbf{s}_i$	Anomaly score of the i^{th} variable.
$\mathbf{a}_i$	Anomaly flag of the i^{th} variable.
E_θ	Encoder network function with θ trainable parameters.
D_ω	Decoder network function with ω trainable parameters.
$\mathcal{G}(\mathcal{V}, \Theta)$	Graph network with nodes in $\mathcal{V}$ and edges Θ parameters.
$\mathcal{D}$	Probability density or mass distribution.
P	Probability function.
$\mathbb{E}$	Expectation operator.
$\mathbb{R}$	Set of all real value numbers.
$\mathbb{Z}$	Set of all integer value numbers.
K	A thousand.
M	A million.
m	Meter.
mm	Millimeter.
cm	Centimeter.
km	Kilometer.
s	Second.
ns	Nanosecond.
kV	Kilovolt.
TeV	Tera electron volt.

Chapter 2

Anomaly Detection and Prediction in Industrial Systems

This chapter discusses machine learning applications for fault monitoring applications and reviews related works in anomaly detection and prediction, transfer learning mechanisms, and anomaly CD methods on time series and spatial-temporal data.

2.1 Time Series Anomaly Detection

An anomaly is a peculiar observation that deviates so much from other observations as to arouse suspicions that a different mechanism generated it. The terms anomaly and outlier are often used interchangeably, but there is a subtle difference between them. Outliers are observations that do not follow the expected behavior–could be noise or rare normal events–whereas anomalies are outliers caused by interesting underlying problematic phenomena [80, 87]. We utilize the terms interchangeably unless we explicitly specify (see Definition 1). Depending on the setting and application domain, anomaly detection methods aim to catch anomalies, abnormalities, deviants, outliers, discords, failures, intrusions, exceptions, aberrations, peculiarities, or contaminants (see Definition 2) [81, 87, 88].

Definition 1. *An anomaly* is an unusual behavior–potentially caused by or able to cause a system fault–with a deviating signature as compared to normal signal patterns.

Definition 2. *Anomaly detection* refers to a method or algorithm for detecting or capturing anomalies on a given observational data. The AD model generates binary status $AD(x) : \Lambda \to \{0; 1\}$ for a given input data x such that:

$$AD(x) = \begin{cases} 0, & if\ x\ is\ normal \\ 1, & if\ x\ is\ anomalous \end{cases} \tag{2.1}$$

Early efforts on AD employ traditional methods, such as local outlier factor (LOF) [89], isolation forest (IF) [90, 91], principal components analysis (PCA) [92], one class support vector machine (OCSVM) [93], and k-nearest neighbor (KNN) [94]. These approaches are ineffective for learning high dimensional data, nonlinear behavior, and temporal dependencies [87]. Deep learning has become effective in capturing complex nonlinear structures, extracting end-to-end automatic features, and scaling for high dimensional and large volume data sets [80,81,87]. Several DL models have been proposed in the literature on diverse data types—including but not limited to structural data [80], time series data [41,52,55,95–108], image data [109,110], graph data [111–115], and spatio-temporal data [110–121].

Time series anomaly detection (TSAD) deals with capturing temporally contextual anomalies. A wide range of algorithms has been proposed in the literature to perform detection of common outlier scenarios on time series data—point-wise detection for a time data point outlier, and pattern-wise detection for subsequences outlier [10,88] (see Fig. 2.1). A point outlier is a datum that behaves abnormally in a specific time instance when compared either to the rest of the time series data—global outliner—or to its adjacent neighboring points—local or contextual outlier. Subsequence outlier—also known as collective outliers and includes shapelet and seasonal outliers—are consecutive points with joint unusual behavior despite each observation individually not necessarily being a point outlier. Subsequence outliers can be categorized into shapelet, seasonal, and trend outliers depending on the distortion scenario. Refs. [11,122] categorize trend anomalies based on the nature of the drift into different types, such as sudden drift, incremental drift, gradual drift, and reoccurring drift (see Fig. 2.2).

Anomaly detection studies can be categorized based on the number of monitored variables into univariate and multivariate AD [88]. Although most of the research studies concentrate on univariate AD, multivariate AD brings richer system information for complex systems from multiple sensors. Detecting anomalies in multivariate time series data is particularly challenging. This is partly caused by the difficulty of capturing potential interaction of the multivariate due to curse-of-dimensionality and heterogeneous temporal characteristics. Two main strategies are often proposed to mitigate this challenge: 1) dimension reduction—e.g., using a PCA, and 2) sequence similarity measuring methods such as dynamic time warping and correlation. These approaches have limited capability in capturing non-linear and inherent characteristics from multivariate signals, which may reduce the efficacy of the AD models. Time series models must consider the complex and diverse nature of temporal data, such as the dimensionality, the stationarity, and the temporal correlations among observations [82]. The recent advances in deep learning deliver relevance in multivariate TS modeling through automated feature extraction [87].

AD paradigms can be categorized into supervised and unsupervised approaches. The supervised approaches treat AD as a classification problem—requiring curated labeled anomaly data to train the model; the unsupervised methods do not require label anomaly data and often involve modeling—trained on—of the healthy data without anomaly contamination and employ nonconformity deviation metrics to detect anomalies during inferencing. The lack of available labeled ground-truth anomaly data and the cost of data annotation limits the pertinence of supervised

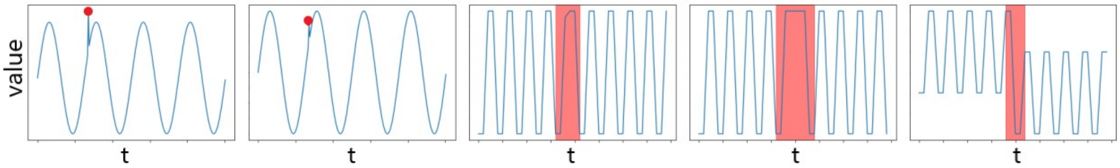


Figure 2.1: Types of temporal anomalies proposed in Ref. [10]: a) global outlier, (b) contextual outlier, (c) shapelet outlier, (d) seasonal outlier, and (e) trend outlier.

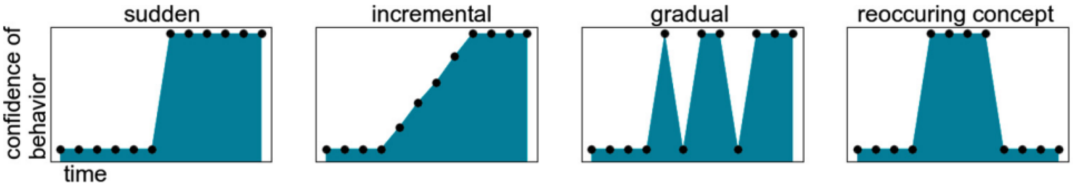


Figure 2.2: Different types of temporal drifts presented in Ref. [11].

AD models in real-world applications despite their superior accuracy. A less popular type of quasi-supervised learning for AD is weakly supervised AD (WSAD), which addresses label data curation challenges by developing AD models with incomplete, inaccurate, and inexact data [123]. Unsupervised approaches are relatively more pragmatic in many real-world settings because of their flexibility—no labeled anomaly data requirement—and leverage for detecting previously unseen anomalies [88, 124]. Unsupervised AD models trained with only healthy observations are also called semi-supervised AD approaches. Recent deep anomaly detection (DAD) approaches are becoming more popular for end-to-end modeling and capturing nonlinearity temporal behavior [95–101,103,108,125–129]. The DAD approaches include clustering, probabilistic, prediction deviation—reconstruction and forecasting—AD paradigms (see Fig. 2.3) [12]. The clustering and probabilistic paradigm measure the similarity of the time series segments on the latent embedding space [41]. The state-of-the-art time series unsupervised DAD generally employs mostly prediction models, such as forecasting [95,99–101,126] and reconstruction paradigms [41,95–98,125] to detect anomalies. High prediction error in forecasting or reconstruction suggests the presence of anomaly (see Fig. 2.3). These approaches can be formulated as:

$$A = \|\mathbf{x}_t - \hat{\mathbf{x}}_t\| > \alpha \tag{2.2}$$

where the $\mathbf{x}_t$ is the target TS data at time t and the $\hat{\mathbf{x}}_t$ is its predicted data—reconstructed or forecasted. The $\|.\|$ is prediction anomaly score and the α is the decision threshold to generate anomaly flag A. The reconstruction paradigms try to recover the information present in the time series input data that may also include some irrelevant information or noise; they may not perform well on certain anomalies [10,12,81,130]. The forecasting paradigm predicts future values from past data points and is relatively robust against noise; but is constrained when data is too complex and highly nonlinear for forecasting [12, 130]. Some studies propose hybrid approaches such as reconstruction and forecasting [95], and reconstruction and latent outlier detection [41] to mitigate the limitation.

Recent works have also attempted generative adversarial networks (GAN) [103, 127, 131, 132]. The GAN consists of two networks operating in an adversarial fashion—generator $\mathcal{G}$ and discriminator $\mathcal{D}$ networks (see Section 3.3.4 for technical details). Anomaly detection using GAN employs either the discrimination capability of the $\mathcal{D}$ or the generation capability of the $\mathcal{G}$ [131, 133, 134]. The generation AD is essentially similar to a reconstruction paradigm, while the discrimination AD follows a direct approach as:

$$A = \mathcal{D}(\mathcal{G}(f(\mathbf{x}_t))) \tag{2.3}$$

where the $\mathcal{G}$ and $\mathcal{D}$ are the generator and discriminator of the GAN, respectively, and the $\mathcal{D}$ determines the anomaly class label for the generated time signal of $\mathcal{G}$ from the latent feature of the input TS data $\mathbf{x}_t$ encoded by function f. Ref. [131] employs a Wasserstein GAN (W-GAN) and autoencoder (AE) for AD on the Airbus's helicopter accelerometer health monitoring. The AD approach follows a reconstruction paradigm using AE, but the decoder network is a frozen generator network of the W-GAN. The authors have trained the W-GAN on the healthy dataset and generator samples from random noise; the AE is trained with a frozen decoder so that its encoder network provides the latent vectors as input random samples to the decoder—generator—for reconstruction. Ref. [134] adopts AnoGAN [133] for AD on EEG time series data—converted to image data. The authors have trained the AnoGAN on normal healthy data and employed reconstruction via the generator to detect anomalies. AnoGAN optimizes the generator's input latent vector using gradient descent to approximate the input and generated data. Generative modeling for large multivariate data is challenging due to the complex temporal correlations present in TS data.

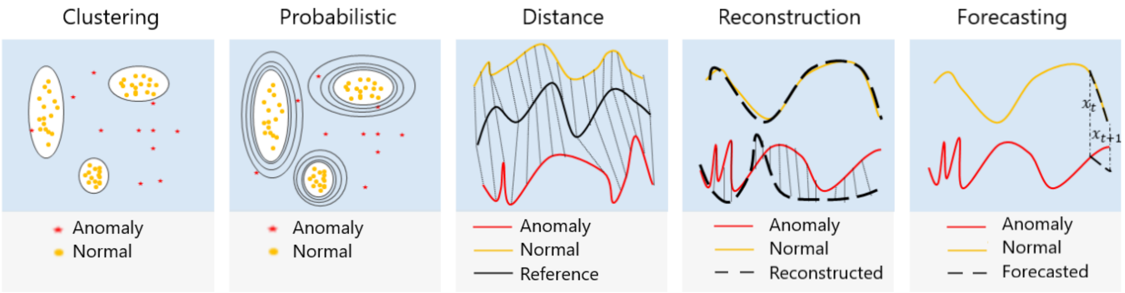


Figure 2.3: Popular unsupervised paradigms of deep learning-based AD approaches for time series data [12].

Refs. [95, 108, 129] propose graph neural networks (GNNs) for AD to enhance learning multivariate data and provide interpretability from graph edge links. Most studies propose correlation-based scores to build the link among the sensors, where top links are selected to reduce the computational complexity [95, 108]. Building a graph network that captures the non-linear relationship among the TS signals remains an open challenge. Ref. [129] attempts to address this challenge and also to avoid the quadratic processing cost of the correlation computation for a graph-based AD approach. Ref. [95] applies graph attention network (GAT) based feature extraction on the variable and time dimension for TSAD. The AD is inferred from reconstruction and forecasted data points through a CNN AE architecture. Ref. [108] presents a graph deviation network (GDN) using a graph attention-based forecasting network for TSAD. The GDN involves trainable sensor embedding layers and building the adjacency matrix by taking the top-K nearest nodes of the dot product of the embedding. They demonstrate that edges in the learned graph provide interpretability by indicating which sensors are related to one another. A similar study by Ref. [129] proposes a convolutional information propagation (IP) mechanism to learn the global topological connections among sensors using a Gumbel-softmax sampling trick and reducing the computation complexity by removing the need for dot product calculation among node embeddings. The IP comprises a multi-scale dilated convolution with residual connection, graph convolutional network (GCN), and transformer to build a time series forecasting (TSF) model for AD application.

Transformers have also found their application in time series AD tasks [135–138]. TranAD [136], MT-RVAE [137], and TransAnomaly [138] propose to combine transformer with generative models, such as VAEs and GANs, while others combine transformer with GNN [129] to enhance reconstruction- and forecasting-based in AD, respectively. Both the MT-RVAE and TransAnomaly combine VAE with transformer networks but for different purposes. TransAnomaly combines VAE with a transformer to allow more parallelization and reduce the training cost by nearly 80%. The MT-RVAE employs a multiscale transformer to extract and integrate time series information at different scales. It overcomes the shortcomings of vanilla transformers, where only local information is extracted for sequential analysis. AnomalyTransformer [135] employs the self-attention mechanism with association discrepancy. It combines transformer and Gaussian prior association to make rare anomalies more distinguishable. The insight is that temporal anomalies are associated with adjacent data points but struggle to have strong associations with the whole data series. The AnomalyTransformer performs weakly and the association learning fails for small time-window sizes, e.g., less than 100. The transformer has quadratic complexity with respect to the window size, and a trade-off is needed for real-world applications.

Ref. [139] explores a self-supervised technique for TSAD inspired by AD on image data from Ref. [140]. The authors convert an unsupervised mechanism into a classification problem using a self-supervision method. The MS2D-Net [139] employs multi-resolution scaling—generates augmented segments with different downsampling rates—for the self-supervision classification on the sudo class labels for each sampling rate.

The research community struggles to provide a consistent and comprehensive overview of their research accomplishments in time series AD [12, 15, 81, 130]. Many exhibit heterogeneity in their methods for AD and runtime evaluation. The lack of adequate well-labeled benchmark data sets to evaluate various anomaly properties—including anomalies that are uni- or multi-variate, point or sequence, unique or repeating, trend, shape and magnitude—make the evaluation and comparison difficult [10, 15, 130]. Most TSAD studies thus focus on particular domains—severely restricting the scope of their study or model evaluations. The data sets in the studies do not adequately cover the different types of anomalies. Refs. [12, 15, 130] have shown that simpler methods perform similarly to more sophisticated deep learning methods. The DL approaches outperform simpler anomalies in detecting global point anomalies and trend drifts—the easiest and the hardest to detect, respectively [12, 130]. The MTAD-GAT [95]—among the deep models—fails in detecting collective anomalies despite being a hybrid of construction and forecasting AD [12]. The one-class SVM easily detects global point and collective shapelet anomalies but still presents decent results for other anomaly types [12].

The recent hybrid DL using variational networks [41, 136, 137] or association learning [135] apprised enhancing TSAD are not adequately covered in the above review studies [12, 15, 130]. The review studies demonstrate that no one-fits-all solution clearly performs best; different algorithms have specific strengths instead.

2.1.1 Spatio-Temporal Anomaly Detection

Spatio-temporal (ST) data contains data points with spatial and time attributes. A unique quality of ST data that differentiates it from other classic data is the presence of dependencies among measurements induced by the spatial and temporal attributes, where data correlations are more complex to capture by conventional temporal or spatial techniques [141]. ST data is available in diverse domains, such as visual streaming data [116–121, 142], transportation traffic flows [111, 112], sensor networks [113–115], geoscience [109], medical diagnosis [110], and high energy physics at the LHC [59].

DAD algorithms have become increasingly prevalent for reliability, safety, and health monitoring in several domains with the proliferation of sensor data [80–82]. AD applications on ST data covers several tasks, such as machinery fault diagnosis and prognosis [143, 144], electronic device fault diagnosis [41, 52, 55, 145], medical diagnosis [96, 106, 110, 146], cyber-security [102, 113, 114], crowd monitoring [116–121, 142], traffic monitoring [111, 112], environment monitoring [109], internet of things [82, 101], and energy and power management [104, 147]. The concept of ST anomaly can be defined as an ST point or cluster of points that deviates from the nominal ST auto-correlation structure of the normal points. Ref. [116] presents an appearance and motion DeepNet (AMDN) approach to address anomalous event detection on video data. The work revolves around fusing learning feature representations and combining appearance and motion information using denoising autoencoders for AD in intelligent video surveillance. Ref. [117] discusses a similar fusion method with sparse coding. Ref. [117] investigates a convolutional ResNet autoencoder with separate data representation encoders for the spatial and temporal dimensions to detect an abnormal event in videos with ST dissociation. Ref. [118] proposes AD in a sequence of image frames in videos using two parallel stacked RNN autoencoders. Temporally-coherent sparse coding is suggested after multi-patched feature extraction using ResNet layers. Ref. [120] devises AD using CNN AE through learning temporal regularity in video sequences. A variational RNN (VRNN) [111] employs graph convolutional network (GCN) for AD on graph traffic flow time series data. The previous works on visual data sets tend to focus on CNN for regular spatial feature extraction, whereas graph-based models are becoming more attractive for ST data that exhibit irregular spatial characteristics, such as traffic data with non-Euclidean distance among spatial nodes.

2.1.2 Online Anomaly Detection

Not all of the variables are monitored by trained ML models in large systems. Various factors contribute to coverage limitations—including curse-of-dimensionality degrading model performance, and preparing time-synchronized curated data from many variables across several subsystems is often challenging. Unsupervised scalable AD approaches are relevant to detect anomalies flexibly on demand on the variables that the trained ML models do not actively monitor [124]. There are many challenges in designing unsupervised and online time series AD [124,148,149]:

1. *Lack of labels*: Data annotation on high-dimensional time series is expensive, whereas unsupervised approaches are a reliable and affordable method for identifying new patterns without the need for supervision.

2. *Concept drift of normal behavior*: AD models need to update their regular behavior considerations when the characteristics of the regularity change.

3. *generalization*: There are different time series anomaly patterns, and it is crucial for AD services to perform efficiently on diverse patterns. Unfortunately, current methods lack generalization for diverse patterns.

4. *efficiency*: The online detection procedure must be completed within a limited time for real-time monitoring and facilitating fault diagnostics on large data sets. Complex deep models—with large time complexity—are often of little use in an online scenario despite their better accuracy, as efficiency is one of the major prerequisites for an online AD.

5. *avoid alert spamming*: AD models should focus on crucial alerts when reporting detected outliers to human operators. Reducing false flags is essential to mitigate the "cry-wolf effect"—operators may ignore the reports when monitoring tools send high false detections.

Supervised AD models are superior in accuracy, but they need labeled data. There is a need for unsupervised online methods with enhanced accuracy, efficiency, and generalization since it is not feasible to label and train on every anomaly scenario [149]. An online unsupervised AD method detects the anomaly on incoming data without prior training. Statistical and light-weight machine learning methods have been shown to outperform deep learning on the unsupervised AD problems [148]—particularly on point and collective anomalies [12, 130]. Conventional statistical models can be quickly adopted for online services, but most have insufficient accuracy in real-world industrial applications [150–152].

Time series signals can be decomposed into different components such as trend, cyclical, and residual components. Different applications may have different expectations in detecting anomalies corresponding to these different components. The residual noise component is dominantly employed for detecting anomalies, and the trend is often used in change-point detection, drift detection, and time series segmentation tasks [153]. Trend drift is a continuous and gradual trend change in time series signals; trend drift detection is a nontrivial task, and domain criteria are often utilized to leverage accuracy. Refs. [11, 122] categorize TS drifts into different types—including sudden drift, incremental drift, gradual drift, and reoccurring drift (see Fig. 2.2). We employ an ensemble of statistical time-domain and frequency-domain outlier detection algorithms for unsupervised time-aware online AD since statistical methods enable faster analysis, training, and prediction times [148]. Our method is an ensemble of temporal outlier detection, trend drift detection, and spectral outlier detection to detect several typical TS outliers.

2.2 Time Series Anomaly Prediction

Inadequate maintenance management methods can reduce the overall productive capacity of equipment by up to 20%, and unplanned downtimes and reactive maintenance in industrial systems incur substantial costs each year [88, 154]. Predictive maintenance (PdM) applications often refer to performing AD, diagnostics, and prognostics system monitoring [155]. PdM—as one of the pillars of industry 4.0—embraces early detection of anomalies that aims at predicting critical anomalies of a system to improve asset availability by actuating early maintenance before major system faults [156]. The term anomaly prediction (AP) [156–159] sometimes interchangeable, used for AD [95, 97, 98, 100, 101, 126, 160]. However, we differentiate the two terms in that the former is to detect an anomaly in given data, while the latter refers to early detection of an anomaly or forecasting an anomaly that may be encountered in the future from early-stage signals [70, 156–158]. AD methods can only detect anomalies after they happen, which leads to information delay and limits preparedness to handle them efficiently [156, 157].

Industries conduct PdM conventionally using statistical tests, rule-based alerts, and preset threshold limits [161]. Owing to technological advancements in sensor and data processing technologies, recent PdM approaches emphasize machine learning approaches to capture intricate hidden patterns [143, 155, 158]. The existing data-driven approaches for PdM revolve around the development of supervised models that aim at specific previously known anomalies or/and rely on feature extraction signal processing tools such as variants of Fourier transform, wavelet transform, statistics, and PCA [143, 156–159, 162, 163]. Efforts on automated feature extraction, via end-to-end deep learning, for prognosis mainly focus on remaining useful time (RUL) estimation [164]. Adopting the above methods for multivariate complex systems is constrained due to high cost or lack of annotation on heterogeneous sensor data. Besides, early signs of anomalies are often not easily seen by experts and are challenging to annotate in large data sets from numerous monitoring sensors. Operational quality-altering anomalies, which do not lead to an ultimate breakdown, are often overlooked. Thus, unsupervised end-to-end methods are essential for anomaly prediction system development. Ref. [159] presents ML for AP by forecasting future time series for mobile networks. The approach covers short sequences—forecast up to a horizon with 16 timesteps—and relies on linear models such as linear regression, PCA, and supervised logistic regression. We propose an unsupervised long-horizon AP mechanism using deep learning models to monitor the high dimensional sensor system of the HCAL [70]. The approach contains multivariate time series forecasting and AD models in a pipeline that are trained with and without anomaly contamination, respectively.

2.2.1 Long Sequence Time Series Forecasting

Based on the number of horizon timesteps, time series forecasting approaches can further be categorized into unit-timestep forecasting [165, 166] and multi-timestep forecasting [70,167]. Further classification as short or long mult-timestep forecasting can be given on the mult-timestep category—considering the length of the horizon. Long sequences are harder to predict. We consider long horizons with a sequence length greater than 100 despite the absence of definite consensus on the length in the literature. Many real-world applications require long sequence TS predictions, such as price forecasting in the stock market [168], e-commerce sell prediction [169], traffic forecasting [170], electricity consumption projecting [168, 169, 171], weather forecasting [168, 171], and system monitoring [70]. The ability to effectively capture long-range dependencies between predictor and target data is essential for long-horizon forecasting models [167].

Long-horizon forecasting approaches employ generally sequence-to-sequence (S2S) autoencoder paradigm using RNN variants [168–172] and transformer [167]. Learning long-range dependencies is a main challenge in long-horizon sequence processing tasks. One key factor affecting the ability to learn such dependencies is the path length of forward and backward propagation in the network. The shorter these paths, the easier it is to learn long-range dependencies [14, 173]. RNN models may exhibit some potential constraints in inference speed due to the recursive step-by-step inferencing [167], and in performance because of deterioration when the length of the input sequence increases [174]. To address these challenges, decoder models with parallel generation are proposed using attention mechanisms [168,171,172,175], multilayer-perceptron [169] and transformer [167]. These approaches operate only with predefined horizons, which limits their scalability. The majority of the existing approaches deal with short horizons—fewer than approximately 40 data points [168–170, 172, 176] except the recent ones [167, 175, 177–182]. Refs. [167, 179] demonstrate the efficacy of transformer encoder and decoder architecture with various long horizons in univariate and multivariate time series data sets. Ref. [177] introduces a mix-hop propagation GNNs with a curriculum learning strategy for the multi-step forecasting task. The curriculum strategy gradually increases the forecasting length to leverage the model training from simpler to harder tasks—short to long-horizon.

Recent studies on multivariate TSF focus on transformers because of the parallel processing and learning capacity [178–181, 183, 184, 184]. Ref. [178] presents temporal fusion transformer (TFT)—a multi-horizon forecasting model with static covariate encoders, gating feature selection, and temporal self-attention decoder—to enhance forecasting and preserves interpretability incorporating global and temporal dependency. Autoformer [179] proposes an auto-correlation mechanism operating as an attention module and devises a simple seasonal-trend decomposition architecture to improve TSF substantially. FEDformer [184] integrates frequency domain with Fourier transform and wavelet transform with attention operation. The frequency mapping is proposed to accomplish a linear complexity by selecting a fixed-size subset of frequency. SSDNet [180] combines transformer with state space models (SSM) to provide probabilistic forecasts. It employs a transformer to estimate parameters of SSM from the temporal data and applies SSM to perform the seasonal-trend decomposition and maintain the interpretable ability. ProTran [181] designs generative modeling and variational inference procedures using a transformer for TSF. Pyraformer [182] employs low-complexity pyramidal attention for long-range time series modeling. The hierarchical pyramidal attention module fosters forecasting with a binary tree following the path to capture temporal dependencies of different ranges with linear time and memory complexity. Ref. [185] reviews recent transformers for time series forecasting applications and highlights their strengths and limitations. The authors show that the prediction accuracy of transformers rapidly deteriorates as input length and network depth increase—limiting the capability of many carefully constructed transformers for complex modeling such as long-term forecasting tasks. Generation for variable horizons—longer than the training horizon—requires training of separate models for each target horizon [167]. It is possible to deploy the transformer models into S2S mode—the predicted time segments are reused to forecast new segments; but, the prediction error would get worse as the forecasting error transverses through the whole network of the encoder and decoder instead of using only the decoder network like the RNN S2S model. The question of building a suitable and multi-layered transformer architecture that increases the model performance is still an open research challenge [185]. Real-world sensor data often contains missing or invalid values—leading to variable-length reading segments; incorporating RNN variant models remains relevant for handling variable length, better generalization, and less susceptible to overfitting in time series data [186].

2.3 Transfer Learning

The effectiveness of deep learning in handling large data sets has caught the attention of both academia and industries in the last decade. Its ability to learn nonlinear behavior along with end-to-end automatic feature extraction allows it to find complex patterns within high-dimensional large data sets. Most deep learning applications require extensive modeling data sets that can be expensive and time-consuming to curate—especially in the case of multivariate time series data. Techniques like transfer learning—adopting pre-trained models into a new task—are potential solutions to develop DL models with limited cleaned data [187–190].

Transfer learning is a paradigm where knowledge from a source pre-trained model on different domains or tasks is utilized as auxiliary knowledge to improve the training efficacy of a target model [191]. One of the common broad categorizations of TL approaches is based on the similarity and difference of the task and domain between the source and target [126, 191, 192]: 1) inductive TL: the source and target tasks are different regardless of their domains, 2) transductive TL: the tasks remain the same but the domains, and 3) unsupervised TL: inductive transferring on different but related tasks with unlabeled data sets. TL can be carried out on 1) model parameters: all or some parameters are transferred from a pre-trained source model, and 2) data: all or part of the source domain data instances are utilized to train the target model. We refer to TL to signify the use of learned network parameters (weights and biases from a source model pre-trained on adequate data sets) on a target model for a related task on a different data set, with or without fine-tuning of the parameters [126]. The target data set may be smaller than the source data set.

Computer vision (CV) and natural language processing applications (NLP) have hugely benefited from TL [191, 193]. The recent successes of GANs and transformers on image and sequential data have ameliorated the adoption of TL methods on several applications [193]. The notable contribution of TL to CV and NLP is particularly in transferring feature extraction networks that are trained on immense data sets with very expensive computation grids. Robustly extracted features simplify the complexity and training cost of the final decision networks while simultaneously enhancing the model accuracy of a target task. We refer to such TL mechanism as *freeze and fine-tune* approach. TL is relatively less explored for TS data like sensor measurement despite the abundant studies in images and language data sets [126, 192]; TS data sets are not readily available or accessible—unlike images and texts on the internet—and are often multidimensional and so diverse that require domain-specific knowledge for data curation and preparation.

Transfer learning on temporal data has been investigated in diverse applications—including machine monitoring [187], electricity loads [188], medical [189], and dynamic systems [190]. Ref. [187] employs freeze and fine-tune TL to improve the training speed and accuracy using pre-trained very deep convolutional networks (VGG) on machine sensor data—converted to images by conducting a wavelet transformation to obtain time-frequency distributions. The initial network layers of pre-trained VGG—the feature extraction networks—are frozen, whereas the lower layers—decision networks—are fine-tuned. Ref. [188] has frozen a temporal learning LSTM model when training only the forecasting fully-connected networks (FC) for univariate electricity data. The authors have demonstrated the effectiveness of TL in learning different TS signal patterns and robustness against noise on small training data sizes. Ref. [189] proposes TL on a deep GRU for multivariate classification for clinical data sets. The authors adopted a GRU model—pre-trained on several classification tasks—to provide generic features for simpler linear logistic regression models on new target tasks. They have shown that models trained with TL outperformed task-specific GRU models and are more robust to the size of labeled data. Ref. [190] investigates the potential of TL on fully convolutional networks (FCN) and residual neural networks (ResNet) for chaotic time series classification in dynamic systems. They trained the models on a given chaotic signal pattern and tested them on different chaotic univariate signals. MU-Net [126] employs TL from univariate FCN U-Net network to multivariate AD task using freeze and fine-tuning TL for classification tasks. They apply pre-trained U-Net to each multivariate input variable for feature extraction when scaling to the MU-Net.

There are a few efforts in adopting TL for spatio-temporal (ST) data [79,194,195]. [194] applies TL for cross-city crowd-flow prediction task, where feature extraction ConvLSTM network of the forecasting model trained on one city is fine-tuned on another city data set. Ref. [79] presents an end-to-end TL framework for cross-domain urban crowd-flow prediction using a deep adaptation mechanism on ConvLSTM networks. The deep adaptation network matches the embedding representations of the source and target domain distributions to learn the transferable features between two domains [79, 196]. Ref. [195] fine-tunes an AE model for cross-city collaborative filtering to conduct chain store site recommendations.

Recent DL models built on hybrids of CNNs [117, 119, 120], RNNs [111, 118], and GNNs [111, 112] have gained momentum for TS and ST data in AD and other data mining applications [79,194,197,198]. Most TL studies thus far focus on feature extraction encoding networks and predominantly on forecasting tasks [196]. We have studied the transferability of CNN, GNN, and RNN networks on both the encoder and decoder networks of an ST AD autoencoder and qualitatively evaluated the effectiveness of the TL on reconstruction and AD tasks.

2.4 Anomaly Causal Discovery and Analysis

We will review the literature on anomaly cause discovery in this section—focusing on causal graph learning for time series data.

In the realm of system monitoring for complex systems, it is imperative to delve beyond predictive or descriptive machine learning tasks to fully comprehend the causal relationship—cause and effect—among different variables and systems [199, 200]. The investigation of causality is a prominent area of interest in diverse fields—including but not limited to IT systems [201, 202], transportation [203], medical science [204, 205], meteorology [199], and social science [206]. The causal discovery (CD) can lead to quicker and more effective fault diagnostics once a problem has been detected. Causal presentation of multivariate is an essential component in root-cause analysis (RCA)—the identification of the underlying root causes and provides an explanation of how the faults are impacting the monitored system [202, 207, 208]. Causality models may answer additional diagnostic questions, such as what would happen if faults occur in particular variables and predicting how specific variables trigger faults in other variables or systems. Causal discovery and analysis are becoming increasingly essential in the industry for identifying the underlying behaviors and evolution of process faults [199, 209].

Anomaly detection and causality analysis on large systems are often highly intractable due to diverse operational configurations, disparate data types, and complex fault propagation mechanisms [200]. The relationship among different variables or components often involves multiple time lags in modern cyber-physical industrial systems. These time lags produce a delay of the fault propagation on the causally connected process variables [210]. Although temporal data provides valuable context that enhances AD and CD, it requires special handling to address the specific challenges often posed by the temporal data characteristics or the complexity of causal processes; reliable CD needs to address the causal process dynamics—including slower data acquisition rate than the underlying rate of changes, missing data, measurement error, non-stationarity, unmeasured confounding factors, and causality heterogeneity and non-linearity—concept drift on the causal relationship [199, 209].

We provide definitions below for commonly used terms in causality studies (see Definition 3 to 9).

Definition 3. *Causal discovery* refers to causal graphs learning from data to build qualitative causal knowledge.

Definition 4. *Causal inference* enables answering causality questions from discovered causal interactions. It often integrates statistical and machine learning methods to answer causal questions from data.

Definition 5. *Causal effect estimation* is a special case of causal inference that answers causal or effect questions.

Definition 6. *Root cause analysis (RCA)* is the process of deducing and understanding the underlying cause of the occurring anomalies or faults.

Definition 7. *Confounders* are unobserved (unmeasured) causal variables that influence two or more variables in the observed data.

Definition 8. *Directed acyclic graphs (DAGs)* are directed (edges with direction) graphs with no directed cycles.

Definition 9. *Partial directed acyclic graphs (PDAGs)* are acyclic graphs with both directed and undirected edges.

Data-driven approaches which include statistical [211–216], information theory [213], and machine learning algorithms [200, 210, 212, 217, 218] have been proposed for CD and RCA. Ref. [213] presents kernel-PCA transformation and symbolic transfer entropy to reduce causality analysis computation for RCA fault diagnosis on multivariate nonlinear variables. The residual data of kernel-PCA detects system faults, and normalized transfer entropy determines the causal pathways from process variables to the residual signal. The proposed method identifies only potential root causes and does not provide causal interactions among TS variables. Ref. [211] proposes convergent cross-mapping to build a causal network for TS alarm data root cause tracing in industrial processes. The approach assumes only deterministic system theory that the data is a chaotic time series generated by a nonlinear deterministic system. Ref. [212] combines multivariate nonlinear chirp mode decomposition with Granger causality to detect and analyze root causes for multiple plant-wide oscillations in a process control system. The approach involves oscillating variable clustering and Granger causality to each group to obtain the root causes. Ref. [200] employs a spatio-temporal pattern network for RCA on TS anomalies in distributed complex systems. The anomalies are detected from changes in the causality dependency network—from a restricted Boltzmann machine model trained on symbolic representation of the healthy TS data. The energy strength of switching or flipping a symbolic pattern indicates potential root causes. Ref. [203] discusses fusing expert knowledge with a data-driven semantic rule mining for adaptive AD and RCA for predictive maintenance of trains. They employ matrix profiling [216]—a sliding time window pattern matching through z-normalized Euclidean distance—to find abnormal discords in sensors and match incoming patterns against those previously confirmed anomalies. Ref. [215] discusses fine-tuning Spearman's rank correlation analysis with domain knowledge rules to build a Bayesian network for fault detection and diagnosis. Ref. [214] utilizes PCA-based fault detection and variable selection using feature importance from extreme gradient boosting for RCA. The study adopts the temporal CD network from [219] to analyze the root cause of faults without historical fault information. [217] employs deep graph convolution neural networks to solve the sparse and nonlinear problem of the SSM for RCA.

Recent approaches have integrated causality inference into the AD deep learning models [210, 218]. Ref. [218] presents an MTCMS framework using temporal convolution and multi-head self-attention networks for CD and a contrastive causal graph for RCA in multivariate time series. They propose modifying the CNN with feature reconstruction and skip connections to improve feature extraction and detect delay-time causalities. The multi-head self-attention employs threshold normalization for quantifiable causal inference. Ref. [210] proposes a sparse causal residual neural network (SCRNN) to extract multi-time-lag causal relations for industrial process fault diagnosis concurrently. The parameters of the SCRNN model describe the integral causal structure by optimizing an MTS forecasting objective with hierarchical sparsity constraints.

Causal graph networks are popular—for their intuitive presentation capability [201, 207, 208, 220]; a causal graph provides a visual presentation of the causality using graphs. The graphs show influence and effect paths, provide strength and direction of leakage, and allow calculation of influence propagation on the paths; these attributes make causal graphs popular for identifying the CD and RCA applications [207, 208]. CloudRanger [201] utilizes the non-temporal Peter-Clark (PC) algorithm for TS data to discover the causal graph between anomalies and identifies root causes through a random walk on a transition matrix. They employ correlation between variable pairs to generate the transition matrix. MicroCause [202] enhances the CloudRanger by inferring the causal graph discovery using the PCMCI algorithm [86]—an extension of the popular constraint-based PC algorithm for time series data. The MicroCause estimates the partial correlation between causally related variables given their parents in the graph to compute the transition matrix for the random walk. WhyMDC [220] identifies root causes from non-temporal data by searching for changes in causality on a given directed acyclic causal graph. CausalRCA [207] employs Shapley value [221] to quantify contributions and rank root causes of a point anomaly score using non-temporal structural causal models. EasyRCA [208] identifies the root causes of collective anomalies in TS data from an acyclic causal graph. The approach utilizes a summary causal graph—without time lag specification—of the TS data in a normal operation. EasyRCA finds the root causes for each group either directly from the graph and appearance time of the anomalies or by comparing causal effects in the normal and abnormal regimes. The study does not address the problem of causal graph discovery, and it assumes that the graph already exists, which is a challenging assumption to hold in several real-world complex systems. The above recent approaches for anomaly RCA either consider non-temporal modeling approaches, requiring the availability of a causal network of the normal operation, or are not adequately optimized for binary anomaly flag data [201, 207, 208, 220].

2.4.1 Causal Graph Discovery Methods

The first step of graph-based causality analysis is to construct the graph causal model of the variables (see Definition 10). Causal graph network discovering methods aim to capture direct dependencies and shared drivers among multiple data variables—graph nodes.

Definition 10. *Graph causal model (GCM)* states that if two variables have an edge in between $X \to Y$ in the directed graph, then (X is a direct cause of Y) that there exist interventions on X that will directly change Y (distribution or value). The edge of GCM between variables X and Y can model 1) a direct causal relationship ($X \to Y$ or $Y \to X$), 2) a causal relationship in either direction (X—Y), and 3) a non-causal association ($X \leftrightarrow Y$) due to external common causes.

Causal discovery studies rely on assumptions to infer underlying causal dependencies from observational data [204, 209, 222]:

- *Causal faithfulness assumption*: data independencies ($X \perp Y | Z$) arise rather from causal structure, not from coincidence or expressed differently. If two variables are independent given some other subset of variables, then they are not connected by a causal link in the graph.

- *Causal sufficiency assumption*: the measured variables include all common causes—there is no unobserved confounding variable.

- *Causal Markov assumption*: all variables are independent of their non-effects—nondescendants in the causal graph conditionally to their direct causes (parents). All the relevant probabilistic information that can be obtained from the system is contained in its direct causes or expressed differently. If two variables are not connected in the causal graph given some set of conditions, then they are conditionally independent. A Markov equivalence class is a set of directed graphs with the same skeleton—encodes the same set of conditional independence—and V-structure ($X \rightarrow Y \leftarrow Z$) regardless of the directions of the remaining edges.

Data-driven causal graph structure learning methods can broadly be categorized into:

- *Constraint-based*: Relies on conditional independence (CI) relationships $X \perp Y | Z$—X independent of Y condition on Z—to infer the causal DAG structure. Some of the popular methods include Peter-Clark (PC) [204], grow-shrink (GS) [223], incremental association (IAMP) [224], max-min parents and children (MMPC) [225], and fast causal inference (FCI) [204]). The PC algorithm in [204, 226] is a popular method that builds a causal graph by adding edges based on CI tests. It learns a partial directed acyclic graph (PDAG) representing the dependencies based on the causal Markov condition and the faithfulness assumption. When there is no latent confounder, two variables are directly causally related—with an edge in between—if and only if there is no subset of the remaining variables conditioning on which they are independent. Three main steps are involved in the PC: 1) identifying the graph skeleton—undirected graph—induced by those CI relations, 2) identifying V-structure, and 3) deriving edge directions. FCI [204] relaxes the causal sufficiency assumption of PC to deal with unmeasured latent variables.

- *Score-based*: Employs optimization search for causal DAG structure to the observed data based on a scoring metric [199,209,227,228]; e.g., hill-climbing search (HC) [229], and greedy equivalent search (GES) [230,231]. These methods explore the space of PDAGs structure classes and minimize a global score—e.g., the *Bayesian information criterion* (BIC) and *Bayesian Dirichlet equivalent uniform prior* (BDue)—using add edge, remove edge, and reverse edge operators to return the optimal structure [230, 231].

- *Hybrid*: Combines the ideas from constraint-based and score-based algorithms to enhance accuracy and computational efficiency; e.g., max-min hill climbing (MMHC) [227] and greedy fast causal inference (GFCI) [232]. MMHC algorithm first builds the skeleton network using CI tests and then performs a Bayesian-scoring greedy HC to orient the edges [227]. GFCI embarks with Fast GES to get a first sketch of the graph rapidly, then uses the FCI constraint-based rules to orient the edges in the presence of potential confounders [232].

- *Functional causal model (FCM)*: is recent CD approach that represents the effect as a function of the causes and independent noise terms [227, 232]; e.g., causal additive models (CAM) [233] and causal generative neural networks (CGNN) [222] and others [199, 209]. FCM captures the asymmetry between cause and effect variables for CD—representing the effect Y as a function of the direct causes X and noise factor ϑ as:

$$Y \leftarrow f_\theta(X, \vartheta; \theta) \tag{2.4}$$

where ϑ is the noise term that is assumed to be independent of X, the function f explains how Y is generated from X, and θ is the parameter set of f. Diverse FCM approaches have been proposed in the literature using regression models, structural equation models, autoregressive models, and neural network models [209, 222, 233]. FCM methods also differ on the data type—continuous, discrete, categorical, and mixed—noise term distribution, and linearity [209].

Structured reviews on learning causality methods on GCM are available in Refs. [199, 209], and a more detailed explanation of structural causal models in a textbook [13]. We limit our discussion to studies related to TS data [20, 86, 210, 218, 219, 228, 234–238].

Causal graph modeling for time series data is a growing area of study in several scientific disciplines [234, 236]. Multivariate TS data become abundant in several real-world domains with the growth of the sensor networks, but finding the causal dynamics in such data is challenging for many reasons: non-linearity of the generating process, data non-stationarity, concept drift over time, varying data rates, and missing data [209]. Time series GCM approaches aim to capture time-lag causality besides addressing the stated challenges of TS data (see Fig. 2.4). The typically utilized TS GCM methods include Granger causality [234, 235], constraint-based [20, 86, 228, 228, 236–238], and machine learning [210, 218, 219].

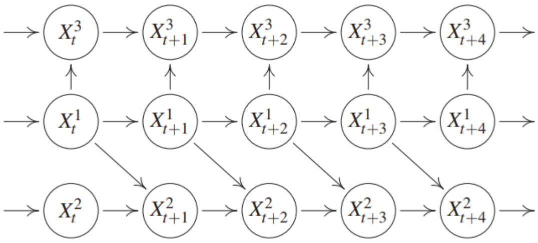


Figure 2.4: A time series with time lag effect $\mathbf{x}_{t-1}^1 \rightarrow \mathbf{x}^2$ and instantaneous effect $\mathbf{x}_t^1 \rightarrow \mathbf{x}_t^3$ [13].

Structural causal model describes a stochastic process $(\mathbf{x}_t)_{t\in\mathbb{Z}}$ from the past q values of all variables:

$$\mathbf{x}_t^i := f^i\left(\left(\mathbf{PA}_q^i\right)_{t-q}, \ldots, \left(\mathbf{PA}_1^i\right)_{t-1}, \left(\mathbf{PA}_0^i\right)_t, \mathbf{e}_t^i\right) \tag{2.5}$$

where $\mathbf{e}_t^i$ is jointly independent noise term. For $s \in \mathbb{Z}^+ : 0 < s \leq q$, the symbol $\left(\mathbf{PA}_s^i\right)_{t-s}$ denotes the set of variables $\mathbf{x}_{t-s}^j$, $\forall j \neq i$, that influence $\mathbf{x}_t^i$.

Granger causality [239] formulates a notion of causality based on how well past values of a time series $\mathbf{y}_t$ could predict future values of another series $\mathbf{x}_t$. Let $\mathcal{H}_{<t}$ be

all the relevant history information up to time $t-1$ and $f(\mathbf{x}_t \mid \mathcal{H}_{<t})$ be the optimal prediction of $\mathbf{x}_t$ given $\mathcal{H}_{<t}$. Granger defined $\mathbf{y}$ to be causal for $\mathbf{x}$ if:

$$\operatorname{var}\left[\mathbf{x}_t - f\left(\mathbf{x}_t \mid \mathcal{H}_{<t}\right)\right] < \operatorname{var}\left[\mathbf{x}_t - f\left(\mathbf{x}_t \mid \mathcal{H}_{<t}/\mathbf{y}_{<t}\right)\right] \tag{2.6}$$

where *var* denotes variance, and $\mathcal{H}_{<t}/\mathbf{y}_{<t}$ indicates excluding the values of $\mathbf{y}_{<t}$ from $\mathcal{H}_{<t}$. Granger assumes the identifiability of a unique linear model with N time series variables as:

$$\mathbf{x}_t^i = \sum_{s=1}^{q} A_s^{ij} \mathbf{x}_{t-s}^j + \mathbf{e}_t^i,\ \forall j \neq i \tag{2.7}$$

where $A^1, \ldots, A^{N-1}$ are $N \times q$ lag coefficient matrices with order $N-1$. The $\mathbf{e}_t$ is a noise or error term with a diagonal or nondiagonal covariance matrix. Granger's equation corresponds to the time series vector autoregressive (VAR) model being treated as a simple causal model without or with contemporaneous causal effects at $t = 0$. Ref. [240] presents nonlinear Granger methods using structured multilayer perceptrons and recurrent neural networks. They combine sparsity-inducing convex group-lasso penalties on the weights that attempt to extract the Granger causal structure by encouraging specific sets of weights to be zero. The predictability characterization of Granger may not directly imply a causal effect of $\mathbf{y}$ on $\mathbf{x}$—improving the prediction of $\mathbf{x}$ does not necessarily mean $\mathbf{y}$ causes $\mathbf{x}$. The effectiveness of the Granger method in deducing causal connections has thus been a subject of ongoing discussion because of the assumption that predictability implies causality, sensitivity to temporal aggregation and subsampling, and unmeasured confounder effects [235]. Nonetheless, the Granger method remains a valuable tool for analyzing TS data and is widely utilized across various domains—including economics, finance, genomics, and neuroscience [235].

Refs. [20, 86, 228, 236, 237] introduce and extend variants of PCMCI—an extension of the PC algorithm [204] leveraged with false-positive cleaning momentary conditional independence (MCI) for time series CD. The PCMCI methods with linear and non-linear conditional independence tests outperform state-of-the-art techniques in causality detection on large TS data sets across a range of research fields [86]. Ref. [228] proposes Latent-PCMCI (LPCMCI) that relaxes the causal sufficiency assumption of PC extends PCMCI to enhance recall CD with unknown latent confounders using FCI. Ref. [237] extends the PCMCI to handle non-stationarity with regime-dependent causal graphs using a time-windowing method. A slightly different approach is adopted in the time-aware PC (TPC); it employs the PC algorithm for TS by considering time delay, bootstrapping, and pruning [238]. The approach proceeds by unrolling the TS data—adding new nodes with time delay tags—and generating DAG by applying a set of conditions: using causality cannot apply backward-in-time to direct edges and weight thresholding to prune the graph when rolling the DAG. The bootstrapping subsamples time window data iteratively to fine-tune the DAG. Advanced ML models have been proposed for causal graph discovery on TS data [219]. Ref. [219] presents a temporal CD framework (TCDF) using the attention scores of the prediction convolutional network. The TCDF consists of k-independent prediction networks based on the temporal convolutional network (TCN), where k is the number of TS. Each network performs CD through the attention mechanism—obtaining causal time delay information through convolutional kernel weight routes. The sequences other than the target TS are fed into the network for the prediction process of a given target. The sequence with a high attention score is the causal sequence of the target sequence. Training prediction models for each variable may constrain the TCDF scalability when the number of variables increases, which is the case in large complex systems.

The above-mentioned factors of TS data can influence the efficacy of the existing causal graph discovery methods. The general problem of estimating the GCM for TS is not close to being solved despite the progress in understanding how to deal with these problems in various use cases [209].

The second stage of CD is parameterizing the links after discovering the causal graph skeleton structure. Probabilistic parameterization methods such as Bayesian networks allow flexible and faster querying for causal inference. A Bayesian network (BN) is a probabilistic graphical model representing variables and their conditional dependencies through a directed acyclic graph [241]. Bayesian Networks are parameterized using conditional probability distributions (CPD)—each node n in the network is modeled as $P(n \mid \mathbf{PA}(n))$ where $\mathbf{PA}(n)$ represents the parents of a node in the network. Bayesian networks represent causal relationships between the variables using CPDs as measures of the causal strength between nodes. Causality modeling with BNs from a given data involves two phases: 1) building the DAG topology structure, and 2) estimating CPD parameters of the DAG. Although parameter estimation is considered a well-studied subject and can be achieved with less computation cost even with limited data availability, learning the DAG structure is more difficult with exponential computational cost as the data and number of variables grow.

Definition 11. *A Bayesian network* is a probabilistic representation of joint distributions over the variables using a DAG model. The CPD is computed using the DAG from a given data using Bayes and chain rules of probability as follows:

$$P(A, B, C) = P(A|B, C)P(B|C)P(C) \tag{2.8}$$

The joint distribution of all variables is the sum of all CPDs in the network as shown by the above equation. Representing the joint distribution's independencies in a graph structure allows storing fewer BN parameters overall. *Maximum likelihood estimation* and *Bayesian parameter estimator* are widely employed BN parameter learning methods [229]. The *maximum likelihood estimation (MLE)* estimates the CPDs simply using the relative frequencies with which the variable states have occurred. The MLE has the problem of overfitting the data—e.g., it will be extremely far off for small data that are not fully representative of the underlying distribution. The *Bayesian parameter estimator* aims to mitigate the overfitting issue of MLE by starting with already existing CPDs using prior histogram counts before the data was observed. For instance, *K2 prior* adds prior initial pseudo-state counts—adds 1—to the actual counts before normalization. Those "priors" are then updated using the state counts from the observed data. Another choice of prior is *BDeu—Bayesian Dirichlet equivalent uniform prior* [229].

Chapter 3

Machine Learning for Time Series Modeling

This chapter discusses time series modeling with machine learning and highlights the working principle and the architecture of deep learning models in the field.

3.1 Time Series Modeling

Time series modeling is a subset of sequence modeling that refers to models and algorithms operating on time series data. A time series data is an ordered collection of pairs of measurements and timestamps. It can be mathematically defined as:

Definition 12. *A time series (TS) data* is matrix data $\mathrm{X} \in \mathbb{R}^{T \times N}$ where the reading data points are stored in sequence and indexed with timestamp t:

$$\mathrm{X} = \mathbf{x_1}, \ldots, \mathbf{x_t} \tag{3.1}$$

where $\mathbf{x} : \{x^i\},\ i \in \{1, \ldots, N\}$, and the N is the number of variables. The size of $N = 1$ determines whether the data is *univariate (UTS)* for $N = 1$ or *multivariate (MTS)* for $N > 1$. The MTS data can be further subcategorized into low, intermediate, and high dimensional based on the size of N or/and M despite there being no clear common consensus in the literature where the boundaries lie; for instance, Ref. [81] uses dimension thresholds of $(1, 10]$, $(10, 50]$, and $[50, -)$ to categorize TS anomaly detection data sets into the above three categories, respectively, whereas Ref. [242] considers high dimensional for TS forecasting data set with a variable size in the order of 100s or 1000s.

Definition 13 provides a generic formulation of a time series model for a given input predictor X and output target Y TS data.

Definition 13. *A time series model* is a sequence model $\mathcal{F}$ that predicts Y from X, $\mathcal{F} : \mathrm{X} \to \mathrm{Y}$:

$$\mathbf{y_{t-S}}, \ldots, \mathbf{y_{t+H}} = \mathcal{F}(\mathbf{x_{t-T}}, \ldots, \mathbf{x_t}) \tag{3.2}$$

where T is the time length of the input, and N is the number of input variables; $\mathbf{y} : \{y^i\},\ i \in \{1, \ldots, M\}$, M is the number of output variables, and S and H are past and future time length of Y, respectively.

The $\mathcal{F}$ may belong to different categories depending on the number of variables and time length. The TS model is designated as UTS or MTS model for $N = M = 1$, and $N \geq 1$ or/and $M \geq 1$, respectively. The UTS models learn only the temporal behavior of a single quantity, whereas the MTS models further capture the interaction of multiple quantities. Considering the time length, a TS model can be a unit-timestep or multi-timestep.

The type of modeling task and application can categorize TS models into classification, regression, forecasting, and clustering [167, 188, 190]:

- Classification: predicts known set of class labels C that $\mathrm{Y} \in \mathbb{Z}$ to input TS X.
- Regression: predicts the values $\mathbf{y_{t\leq 0}} \in \mathbb{R}^{S\times N}$ from X.
- Forecasting: predicts the future values of $\mathbf{y_{t>0}} \in \mathbb{R}^{H\times N}$ from X.
- Clustering: measures similarity among the input TS segments X and maps them into different clusters, $\mathrm{Y} \in \mathbb{Z}$.

High-level TS applications—including anomaly detection, anomaly prediction, generation tasks, and causality analysis—can be achieved by employing one or more of the above-defined tasks [41, 70]:

- Anomaly detection: predicts anomaly status label at $t = 0$ from a given TS segments $\mathbf{x_{t\leq 0}} \in \mathbb{R}^{T\times N}$.
- Anomaly prediction: predicts anomaly status label at $t > 0$ from a given TS segments $\mathbf{x_{t\leq 0}} \in \mathbb{R}^{T\times N}$.
- Generation: generates Y from latent space embedding of X.
- Causality analysis: measures causal relationship and dependencies among the X variables.

There are various categories for TS models based on the techniques used for their training and inferencing. The models may employ supervised, semi-supervised, and unsupervised approaches during model training based on varying degrees of utilization of data labeling—full, partial, and no labels, respectively. There are three categories of model training mechanisms: one-time learning, incremental learning—e.g., reinforcement learning, and transfer learning—is a one-time training on source data followed by fine-tuning on another target data.

Machine learning gains attraction for the learning capacity of complex patterns and nonlinear characteristics. Conventional ML models often require feature preparation through selection and engineering before training; this limits their scalability on a range of applications because of their heavy reliance on domain knowledge. The recent progress of deep learning (DL) enhances the learning capacity with flexible and deeper networks (more layers) and offers end-to-end modeling that enables automatic feature extraction. Deep learning with recurrent neural networks (RNN), convolutional neural networks (CNN), and their hybrids are popular for end-to-end learning on sequential data [41, 188, 190]. Transformer and graph neural networks (GNN) present recently promising capabilities that alleviate some of the limitations of RNN and CNN models [108, 167, 177]. Transformers enable parallel processing with an attention mechanism for time series modeling; GNNs are mostly employed to capture interactions of variables from multivariate data. Deep learning networks are packaged in different architectures—including feed-forward architecture, autoencoder (AE), sequence-to-sequence (S2S), generative adversarial network (GAN), and many more consisting of hybrid architectures—to build the final model for a given application.

Numerous methods have been proposed in the literature for TS modeling; the above discussion is thus not meant to be inclusive but rather aims to highlight the widely used methods. We will describe various commonly utilized DL networks and architectures for TS modeling in the following sections.

3.2 Deep Learning Networks

Neural networks—inspired by biological neurons—are the foundation of deep learning networks. The *universal approximation theorem* states that a two-layered feed-forward neural network (FNN) $\mathcal{F}_{fnn}$—also known as multi-layer perception (MLP)—can approximate any arbitrary continuous function given sufficient neurons in the hidden layer [243]:

$$\begin{aligned} \mathbf{y} &= \mathcal{F}_{fnn}(\mathbf{x}, \mathbf{W}, \mathbf{b}) \\ \mathbf{y} &= \sigma_o\left(\mathbf{W_o}(\sigma_h\left(\mathbf{W_h}\mathbf{x} + \mathbf{b_h}\right)) + \mathbf{b_o}\right) \end{aligned} \tag{3.3}$$

where $\mathbf{x} \in \mathbb{R}^{N_x}$ and $\mathbf{y} \in \mathbb{R}^{N_y}$ are the input vector and output variables with dimension of N_x and N_y, respectively. The $\mathbf{W_h} \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{W_o} \in \mathbb{R}^{N_y \times N_h}$, $\mathbf{b_h} \in \mathbb{R}^{N_h}$, and $\mathbf{b_o} \in \mathbb{R}^{N_y}$ are the weights and bias, trainable network parameters of the neurons, for the hidden and output layers, respectively. The σ is the activation function (linear or nonlinear).

The activation function determines which information should be transmitted to the next neuron. Nonlinear activation functions theoretically aim to enable neural networks to approximate any function—ensuring the outputs are beyond the linear combination of the inputs [244]. The range and continuously differentiable attributes are also important properties of activation functions. A finite range makes the training gradient more stable, but an infinite range uses weights more efficiently. Differentiable activation functions are beneficial for gradient-based model training. Fig. 3.1 illustrates some of the widely used nonlinear activation functions, such as *sigmoid function* (sigmoid), *hyperbolic tangent function* (tanh), rectified linear unit (ReLU), *leaky ReLU* (LeakyReLU), *parameterized ReLU* (PReLU), and *exponential linear unit* (ELU) [245]. Each of the above functions has strengths and weaknesses. For instance, the sigmoid, and tanh have bounded outputs (0, 1) and (-1, 1), respectively, and are easily continuously differentiable but may have vanishing gradients and slow convergence. The ReLU greatly accelerates the convergence and makes the network sparse by not activating all the neurons at the same time; but, it is non-differentiable at zero—causing the gradients for negative inputs to become zero, which hinders updating the weights in the region during backpropagation. The LeakyReLU, ELU, and PReLU aim to achieve continuously differentiable and solve the dying ReLU for negative inputs. The LeakyReLU uses ρx, where ρ is a tunable hyperparameter for small negative inputs; but, it still saturates for large negative values. The ELU offers a generalization of the ReLU with a parameterized exponential function for small negative values that push the mean of the activations closer to zero to boost training speed. The PReLU is also a generalization of ReLU with an adaptive control parameter (w_ρ) that is tuned during model training. There are other less known but promising activation functions such as Swish [246] and Misra [247] with demonstrated improved performance in gradient flow, accuracy, and generalization. The activation functions make sigmoid and tanh nonboundary using $f(x) = x.sigmoid(\beta x)$ and $f(x) = x.tanh(softplus(x))$, where $softplus(x) = ln(e^x + 1)$, respectively.

Activation of the output layers (σ_o) highly depends on the target variables and the model applications—for example, sigmoid for categorical output in binary classification and ReLU for continuous output in regression. The choice of activation function also depends on the data set range or scale—for instance, using tanh in the output activation would be a better option than ReLU for a regression task on continuous target data with the range of $[-1, 1]$. Deep learning generally requires data scaling for effective model training, considering the employed activation functions. Some of the typically employed data scaling techniques are data normalization—e.g., *min-max* into $[0, 1]$ and $[-1, 1]$, and standardization—e.g., *z-scale* into $mean = 0, std = 1$ (see Section 3.5 for further discussion).

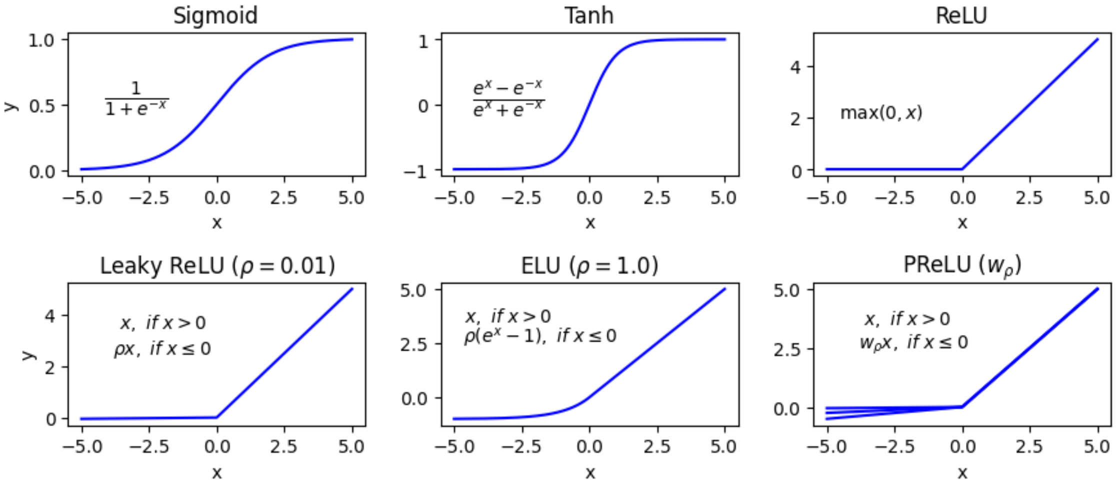


Figure 3.1: Non-linear activation functions.

Learning algorithms find the best parameters during neural network training for the $\mathbf{W}$ and $\mathbf{b}$ that map a set of inputs to their correct output by minimizing loss function $\mathcal{L}(\mathbf{y}, \mathbf{y}_a)$. The $\mathcal{L}$ measures the discrepancy between the target output $\mathbf{y}_a$ and the computed output $\mathbf{y}$. Optimization algorithms like *gradient descent (GD)* are employed to minimize the loss—also called objective or cost—function. GD calculates the opposite steepest descent using first-order derivation for finding a local minimum through *backpropagation* and iterative optimization. The motivation for backpropagation is to calculate the GD of the objective function $\Delta\mathcal{E}_{w_{ij}}$ with respect to parameters using a *chain rule* as:

$$\Delta\mathcal{E}_{w_{ij}} = \frac{\partial\mathcal{E}}{\partial w_{ij}} = \frac{\partial\mathcal{E}}{\partial\mathbf{y}}\frac{\partial\mathbf{y}}{\partial w_{ij}} = \frac{\partial\mathcal{E}}{\partial\mathbf{y}}\frac{\partial\mathbf{y}}{\partial\sigma}\frac{\partial\sigma}{\partial w_{ij}} \tag{3.4}$$

where $\mathcal{E} = \mathcal{L}(\mathbf{y}, \mathbf{y}_a)$, and the σ is a differentiable activation function. Using a similar formulation, the backpropagation through the chain rule enables calculating the gradient on a multi-layered deep neural network. Finally, each training iteration updates network parameters following the direction of loss-minimizing GD as:

$$\begin{aligned} \Delta w_{ij} &= -\lambda\frac{\partial\mathcal{E}}{\partial w_{ij}} \\ w_{ij}^{\text{new}} &= w_{ij} + \Delta w_{ij} \end{aligned} \tag{3.5}$$

where is λ is the learning rate.

One of the key innovations in deep learning is expanding the model complexity by adding more network layers that leverage the network learning capacity on large data sets. Training DL is a computationally intensive task, and stacking more layers may lead to a quick reduction in accuracy due to overfitting, gradient vanishing or explosion, and internal covariate shift. Overfitting occurs when the machine learning model memorizes the training data instead of learning the general context—the model consists of more parameters than that data needs. Deep learning models may suffer from gradient vanishing and explosion during training since the gradient's partial derivative multiplication chain grows with the network depth [248, 249]. The internal covariate shift phenomenon refers to the changes in the input data distribution—means and variances—as it passes through the network layers during training [249]. The randomness in the network parameter initialization and the randomness in the input data contribute to this shift. Deeper networks amplify the covariate shift as the shift propagates through multiple layers. Internal covariate shift slows down the training by requiring lower learning rates and careful parameter initialization and makes it notoriously hard to train models with saturating nonlinearities [249].

Several mitigation techniques have been proposed to address the above challenges. Model regularization techniques—including weight regularization, dropout, and early-stopping—are among the popular methods for preventing overfitting. Network weight regularization limits the value of the network parameters from growing larger. A regularization term is explicitly added to optimize the model training loss function.

$$\mathcal{L}_r = \mathcal{L} + \lambda_w \left\|\mathbf{W}^2\right\| \tag{3.6}$$

where the $\mathcal{L}_r$ is loss function with $l_2 - norm$ weight regularization with factor of $\lambda_w \leq 1$. Dropout is another technique that limits the number of active network weights during the training iteration to prevent overfitting [250, 251]. It randomly prunes weights (set zero) that make each layer receive input only from a random subset of nodes in the previous layer:

$$\bar{w}_{ij} = \begin{cases} w_{ij}, & \text{with } P(c) \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

where w_{ij}, and $\bar{w}_{ij}$ are the weight before and after weight dropout, respectively, and the $P(c)$ probability of keeping a weight. Early-stopping follows a different approach in which the model training is halted before its performance drifts when evaluated in separate validation data sets. The concept is that the training loss keeps dropping as the model overfits the training data while the loss increases on unseen validation data; the training is early-stopped when overfitting begins to occur by measuring performance disparity on the training and validation data sets.

Residual learning successfully enables training several popular models with considerable depth—e.g., the VGG model with 152 layers [252]—and complex models—e.g., transformers [14] by preventing gradient vanishing. Residual learning eases the training by explicitly reformulating the layers as learning residual functions (with skip connections) with reference to the layer inputs:

$$\mathbf{X}^{l+1} = \mathcal{F}^{l}(\mathbf{X}^{l}) + \mathbf{X}^{s} \tag{3.8}$$

where $\mathcal{F}^{l}$ and $\mathbf{X}^{l}$ is the network function and input data at l^{th} layer, respectively. The $\mathbf{X}^{\mathbf{s}}$ is a residual skip connection from the previous layer $s^{th} \leq l$. The residual learning addresses the vanishing gradient problem as the gradient of training loss—$\mathcal{E}$ with respect to the $\mathbf{X}^{l+1}$—will have the term $\frac{\partial \mathcal{E}}{\partial \mathbf{X}^{l}}$ that prevents gradient vanishing as depth grows. ResNet [252] introduces residual learning for CNN models on image applications; the residual concept is not new to TS modeling with RNNs [248]. LSTM [248] addresses the vanishing gradient problem of RNNs with weighted residual connections—forget gates—enabling training very deep RNNs with a very long time span.

We have employed batch normalization (BN) [249]—to mitigate internal covariate shifting and accelerate train model training with more stable data distribution between layers—for our models in Refs. [41, 70, 77, 78]. BN achieves normalization by re-centering and re-scaling layer inputs at a batch level as:

$$y = \gamma \frac{x - \mu_x}{\sqrt{{\sigma_x}^2 + \epsilon}} + \beta \tag{3.9}$$

where the mean $\mu_x = \mathbb{E}[x]$ and variance ${\sigma_x}^2 = \mathrm{Var}[x]$ are calculated per dimension over the training mini-batches, and γ and β are learnable parameter vectors of size N_x. The ϵ is an arbitrarily small constant added in the denominator for numerical stability. The BN substantially improves training convergence speed and lessens the impact of random network initialization schemes. It enables the usage of higher learning rates by smoothing gradient—alleviating vanishing or exploding gradients [253]. BN technique has a regularizing effect on the network—improving its generalization properties and reducing the need for dropout to prevent overfitting. Some studies have argued that the performance returns of the BN is rather by smoothing the optimization landscape of the cost function, and its contribution in internal covariate shift mitigation is limited [253]. Ref. [254] demonstrates deep BN networks—BN in multiple deep layers—causes exploding gradient signals that grow exponentially with depth and the issue can be mitigated using residual connections.

Attention mechanisms—another major progress in deep learning—enhance the effectiveness of automatic searching or capturing of relevant parts from the input data for predicting the target [14, 255, 256]. Attention [255] mitigates the fixed-length vector bottleneck of language translation capability of an LSTM sequence-to-sequence (S2S) model of [256]. Ref. [14] kindles a major revolution using multi-head attention in their transformer models for large-scale natural language processing (NLP).

We will explain below the working mechanism of DL networks that are prevalent in TS classification, forecasting, and anomaly detection applications.

3.2.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are the earliest and most utilized endeavors of neural networks to handle sequential data through sequential learning mechanism [88, 257–261]. RNNs have loops that enable sequential information to persist by using previous state outputs as inputs (see Fig. 3.2).

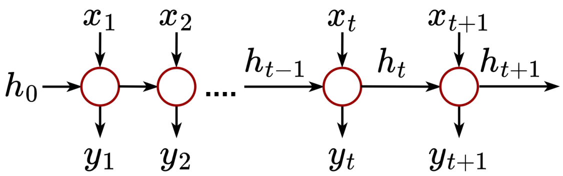


Figure 3.2: Sequential data modeling using vanilla RNN.

For each element in the input sequence, each cell of RNN computes the following function:

$$\begin{aligned}\mathbf{y}_t &= \mathcal{F}_{rnn}(\mathbf{x}_t, \mathbf{W}, \mathbf{b}, \mathbf{h}_{t-1})\\ \mathbf{y}_t &= \mathbf{h}_t = \sigma\left(\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}\right)\end{aligned} \tag{3.10}$$

where $\mathbf{x}_t$ and $\mathbf{y}_t$ are the input and output at t, respectively, and $\mathcal{F}_{rnn}$ is RNN model. The $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$ are the hidden states at t and $t-1$, respectively. The $\mathbf{W}_{\mathbf{i}*} \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{W}_{\mathbf{h}*} \in \mathbb{R}^{N_h \times N_h}$, and $\mathbf{b} \in \mathbb{R}^{N_h}$ are weight matrices and bias vector parameters with the superscripts N_x and N_h refer to the number of input features and the number of hidden units, respectively. The σ is the activation function, such as sigmoid $\in (0, 1)$, or tanh $\in (-1, 1)$.

The RNN architecture offers several benefits—including processing input of any length, computation considering past information, and shared weights across time. Sluggish computation and vulnerability to vanishing and exploding gradient phenomena when learning long-time sequence dependencies constrain the application of RNNs. The gradient issues occur due to the multiplicative gradient during backpropagation, which substantially decreases or increases with the size of the time length. Ref. [262] devises *gradient clipping* technique to deal with the exploding gradient problem by capping the maximum value for the gradient. The long short-time memory networks (LSTM) [248]—an enhanced variant of RNNs— remedy the vanishing gradient problem. Ref. [174] presents the gated recurrent unit networks (GRU) with fewer model parameters by simplifying LSTM. Both LSTM and GRU basically employ specific RNNs combined with well-defined gates (Γ) that enable enhanced control on the information reservation cross time:

$$\Gamma = \sigma_g \left(\mathbf{W}_i \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b} \right) \tag{3.11}$$

where $\mathbf{W}_i \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{W}_h \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{b} \in \mathbb{R}^{N_h}$ are trainable coefficients specific to the gate and σ_g is the activation functions of the gate.

LSTM contains three gates—the forget gate, the input gate, and the output gate. The first gate $\mathbf{f_t}$ determines whether to remember or discard the information from the previous timestamp. The second gate $\mathbf{i_t}$ learns new information from the input, while the third gate $\mathbf{o_t}$ passes updated information to the next timestamp. Thus, the LSTM cell generates short-term and long-term memory state vectors at t—the hidden state $\mathbf{h_t}$ and the cell state $\mathbf{c_t}$, and is formulated as:

$$\begin{aligned}
\mathbf{i_t} &= \sigma_g \left(\mathbf{W_{ii}} \mathbf{x_t} + \mathbf{W_{hi}} \mathbf{h_{t-1}} + \mathbf{b_i} \right) \\
\mathbf{f_t} &= \sigma_g \left(\mathbf{W_{if}} \mathbf{x_t} + \mathbf{W_{hf}} \mathbf{h_{t-1}} + \mathbf{b_f} \right) \\
\mathbf{o_t} &= \sigma_g \left(\mathbf{W_{io}} \mathbf{x_t} + \mathbf{W_{ho}} \mathbf{h_{t-1}} + \mathbf{b_o} \right) \\
\mathbf{c_t} &= \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \sigma_c \left(\mathbf{W_{ic}} \mathbf{x_t} + \mathbf{W_{hc}} \mathbf{h_{t-1}} + \mathbf{b_c} \right) \\
\mathbf{h_t} &= \mathbf{o_t} \odot \sigma_h \left(\mathbf{c_t} \right)
\end{aligned} \tag{3.12}$$

where the $\mathbf{W_{i*}} \in \mathbb{R}^{N_h \times N_x}$, $\mathbf{W_{*h}} \in \mathbb{R}^{N_h \times N_h}$, and $\mathbf{b} \in \mathbb{R}^{N_h}$ are weight matrices and bias vector parameters with the superscripts N_x and N_h refer to the number of input features and the number of hidden units, respectively. The σ_g (e.g., sigmoid), σ_c (e.g., tanh), and σ_h (e.g., tanh) are the activation functions of the gates, cell states, and hidden states, respectively. The $\odot$ denotes the element-wise Hadamard product.

GRU simplifies LSTM to reduce the number of parameters and memory costs while preserving the merits of LSTM. It consists of three gates—the reset gate $\mathbf{r_t}$, the update gate $\mathbf{z_t}$, and the new gate $\mathbf{n_t}$:

$$\begin{aligned}
\mathbf{r_t} &= \sigma_g \left(\mathbf{W_{ir}} \mathbf{x_t} + \mathbf{W_{hr}} \mathbf{h_{t-1}} + \mathbf{b_r} \right) \\
\mathbf{z_t} &= \sigma_g \left(\mathbf{W_{iz}} \mathbf{x_t} + \mathbf{W_{hz}} \mathbf{h_{t-1}} + \mathbf{b_z} \right) \\
\mathbf{n_t} &= \sigma_n \left(\mathbf{W_{in}} \mathbf{x_t} + \mathbf{r_t} \odot \left(\mathbf{W_{hn}} \mathbf{h_{t-1}} + \mathbf{b_n} \right) \right) \\
\mathbf{h_t} &= \left(1 - \mathbf{z_t} \right) \odot \mathbf{n_t} + \mathbf{z_t} \odot \mathbf{h_{t-1}}
\end{aligned} \tag{3.13}$$

where the σ_g and σ_n are the sigmoid and the tanh function, respectively.

There are several modifications of LSTM and GRU in the literature that aim to either improve performance or extend the applications [255, 263–265]. Ref. [263]

finds the contribution of the reset gate of GRU significantly overlaps with the update gate, and proposes a light-gated recurrent unit (LiGRU) that removes the reset gate, replaces tanh with the ReLU activation and applies *batch normalization* (BN). The modification has led to a more efficient and compact single-gate GRU model while the change in activation function coupled with the BN alleviates numerical issues when learning long-term dependencies. Ref. [264] introduces a time-aware LSTM (T-LSTM) that extends LSTM to handle irregular time intervals by introducing time decaying to discount the cell memory content according to the elapsed time gaps. Ref. [255] proposes an attention mechanism to enhance the language translation capability of an LSTM-based sequence-to-sequence model [256]. Ref. [265] presents convolutional LSTM (ConvLSTM) that extends LSTM for learning spatiotemporal data, where the convolution and LTSM handle spatial and temporal learning, respectively.

3.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) have achieved promising performance in several time series applications, and their leverage is often owning to the shared nature of the kernels, also known as filters [41, 70, 95, 128, 177, 187, 190, 242, 266–268]. The convolution refers to multiplying the input neurons with a set of kernel weights [244]. The kernel acts as a sliding window across the whole data, enabling the network to learn features from neighboring data points. The same filters are applied throughout the input data points within a layer—parameter sharing [269]. Parameter sharing enables position-invariant feature extraction and substantially reduces the number of trainable parameters compared to fully-connected neural networks. In TS modeling, 1D CNN handles the sequential data inputs, and it operates basically the same as the classic 2D CNN, except it runs the convolution across the time dimension. Some studies have proposed transforming the TS into 2D image data, then applying the 2D CNN [134, 187]. For a given input TS data $\mathrm{X} \in \mathbb{R}^{n \times N_x}$ with n samples and N_x variables, 1D CNN generates the output $\mathrm{Y} \in \mathbb{R}^{n \times N_y}$ using N_y kernels with window size of T_k as:

$$\mathrm{Y}(:, j) = b(j) + \sum_{d=1}^{N_x} \mathbf{W}(:, j) \star \mathrm{X}(:, d) \tag{3.14}$$

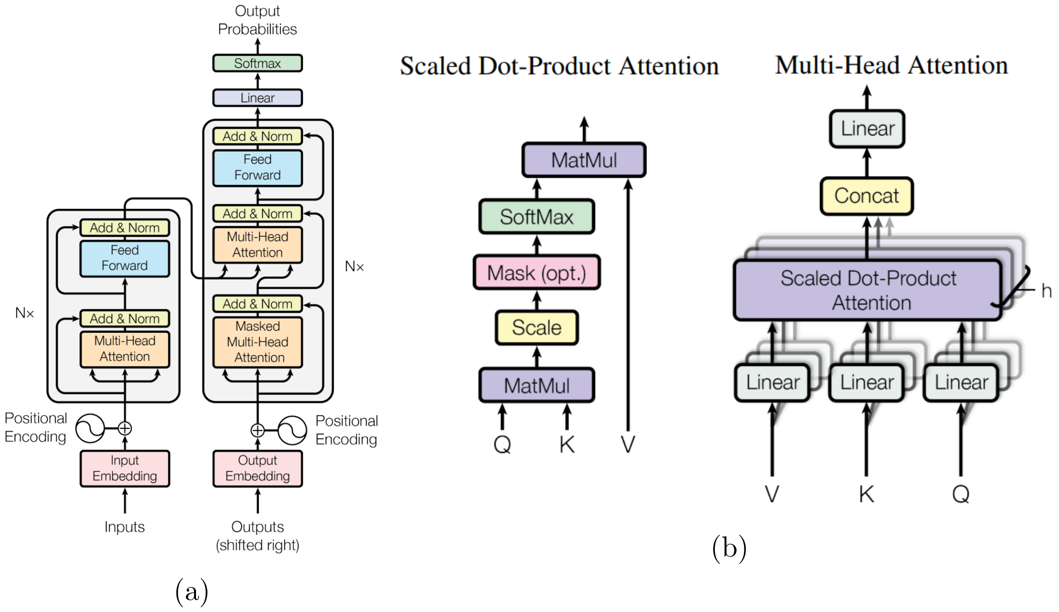
where $\mathbf{W} \in \mathbb{R}^{T_k \times N_k}$ and $\mathbf{b} \in \mathbb{R}^{N_k}$ are the weight matrix and bias vector of the kernels, respectively. The $\mathbf{W}(:, j)$ and $\mathrm{Y}(:, j)$ are the j^{th} kernel and its corresponding output vector. The $\star$ is a 1D cross-correlation operator. The kernels operate on consecutive N_k data points and, thus, extract local behavior. The same kernels are applied across the multivariate data, meaning extracting similar temporal features.

CNNs are often designed with multi-layer stacks in which the different layers learn different contexts: higher-layer for wider context and deeper-layer for distilling features. CNNs require very deep networks with downsampling or extremely large filters to effectively cover a long time length [244, 270, 271]. Temporal convolutional networks (TCN) [266, 270] employ multilayer dilated CNNs to capture the more elongated temporal characteristic. Dilated CNNs have an exponentially large receptive field with a progressing dilation factor d that applies kernel by skipping some d adjacent data points in the sequence; the dilated convolution reduces to a regular 1D convolution for $d = 1$. Employing larger dilation permits an output to represent a wider range of inputs—effectively expanding the receptive field of the network [270]. We can adjust the receptive field size of CNNs—the number of layers, dilation factors, and kernel sizes—to control the model complexity for different domain requirements.

CNNs are relatively easier to explain in contrast to RNNs since each filter kernel can be visualized to assess the temporal characteristics they are responding to. There are certain advantages over RNNs apart from the simplicity: 1) lower model parameters to train because of kernel weight sharing, 2) advantage of parallelism when performing convolutions, 3) localized feature extraction aids to reduce temporal dimension using pooling mechanisms, 4) the gradients are not in the temporal direction but in the direction of the network depth, which can be handy for longer time length, and 5) the gradients are relatively stable—particularly with the residual connections. The major drawback is during model inference: CNNs may require more data storage than RNNs since it takes the entire time window sequence segment; RNNs keep only the summary in the form of a hidden state and need only to process a single time step at a time [271]. We refer the readers to Ref. [244] for further review of recent modifications, applications, and limitations of the CNNs.

3.2.3 Transformers

The recent development of transformer models combines the contribution of the previous achievement of attention, residual learning, and normalization techniques [14]. The notable innovation is the application of multi-head attention for parallel processing of sequential data. Transformers have excelled in natural language processing tasks and become promising in other applications—including computer vision [272], TS audio systems [273], and other TS applications [185]. Among the many benefits, capturing long-term dependencies while achieving parallelization is particularly appealing for TS modeling—resulting in exciting progress in various applications [138, 167, 184, 274–276]. Transformers are DL models that employ parallel processing and the self-attention mechanism to weigh the importance of each data point in a sequence of data points [14] (see Fig. 3.3); the attention mechanism captures the temporal context for any position in the input sequence while processing in parallel. Transformers employ position encoding mechanisms—e.g., sinusoidal encoding often used to encode time stamps—to encode the position of the data points in the sequence [167]. Transformers have much higher bandwidth since they can directly attend to every data point in the input sequence; RNNs have constrained sequential access. The parallelization enables training on much larger data sets. The recent massive pre-trained models—like the generative pre-trained transformer (ChatGPT)—were trained on massive text corpus data sets in the size of terabytes. Several variants of transformers have also been proposed to address challenges in various TS modeling tasks, such as forecasting [167,184,274,277], classification [138, 278], anomaly detection [129, 135, 136, 276, 279], and representation learning [280].



(a)

(b)

Figure 3.3: Transformer network architecture of a) transformer encoder model architecture, b) (left) dot product attention, and (right) multi-head attention of several attention layers running in parallel [14].

The vanilla S2S transformer follows an encoder-decoder stack architecture (see Fig. 3.3). The transformer comprises self-attention mechanisms and feed-forward neural networks in its encoder and decoder. The self-attention mechanism takes the encoded input from the preceding encoder and weighs the importance of each encoding in relation to the others to produce the output encodings. Each output encoding is then individually processed by the feed-forward neural network. The first encoders are input token—word—and token position encoding networks and generate sequences of token embedding X. The transformer model learns three weight matrices for each attention unit—the query weights $\mathbf{W}_Q$, the key weights $\mathbf{W}_K$, and the value weights $\mathbf{W}_V$. The input embedding $\mathbf{x_i}$ for each token i is multiplied with each of the three weight matrices to produce a query vector $\mathbf{q_i} = \mathbf{x_i}\mathbf{W}_Q$, a key vector $\mathbf{k_i} = \mathbf{x_i}\mathbf{W}_K$, and a value vector $\mathbf{v_i} = \mathbf{x_i}\mathbf{W}_V$. The attention weights a_{ij} are calculated from token i to token j is the dot product between $\mathbf{q_i}$ and $\mathbf{k_j}$. The output of the attention unit for token i is the weighted sum of the value vectors of all tokens weighted by the a_{ij}—the attention of i token to each token. The attention calculation for all tokens can be expressed as one matrix calculation among the matrices $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$, where rows are $\mathbf{q_i}, \mathbf{k_i}$, and $\mathbf{v_i}$, respectively:

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{N_k}}\right)\mathbf{V} \tag{3.15}$$

where the $\mathcal{A}$ is dot-product (multiplicative) attention. The N_k is the dimension of the key vector, and the $\sqrt{N_k}$ stabilizes the gradient during training and passes through a normalizing softmax. Separating the $\mathbf{Q}$, and $\mathbf{K}$ matrices—$\mathbf{W}_Q$ and $\mathbf{W}_K$ are separate FC networks—allows the attention to be non-symmetric: if token i attends to token j—$\mathbf{q_i} \cdot \mathbf{k_j}$ becomes large—and this does not necessarily mean that token j will attend to token i—$\mathbf{q_j} \cdot \mathbf{k_i}$ becomes small. Inspired by the multiple kernels in CNN for learning different characteristics, transformers also employ multi-head attention:

$$\begin{aligned} \mathcal{A}_{\text{multi}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \left[\mathcal{A}^1 \parallel \mathcal{A}^2, \ldots, \parallel \mathcal{A}^h\right] \mathbf{W}^O \\ \mathcal{A}^i &= \mathcal{A}\left(\mathbf{W}_Q^i \mathrm{X}, \mathbf{W}_K^i \mathrm{X}, \mathbf{W}_V^i \mathrm{X}\right) \end{aligned} \tag{3.16}$$

where the $\mathcal{A}^i$ is i^{th} attention head, the $\mathbf{W}^O$ is output final projection network, and the $\parallel$ is data concatenation operator. Parallel processing of attention heads simultaneously can speed up the processing of the multi-head $\mathcal{A}_{\text{multi}}$.

In time series modeling, the embedding layers of the transformers are mainly modified to handle TS data—embedding of TS data using 1DCNN and encoding of the timestamp using a circular positional encoding mechanism. The biggest challenge of vanilla transformers is the quadratic complexity $O(n^2)$ of the self-attention, where the n is the dimension of input data dimension. which requires a quadratic memory depending on the input time dimension [185, 281]. In a recent study on time series, Reformer [282] presents the local-sensitive hashing (LSH) attention, and Informer [167] extends the transformer with Kullback-Leibler divergence and probabilistic sparsing mechanism to reduce from quadratic complexity of $O(n^2)$ the self-attention to $O(n \log n)$ for long sequence multivariate time series forecasting application. Both attempt to improve the self-attention mechanism of the vanilla transformer to a sparse version following the point-wise dependency and aggregation. Autoformer [185] proposes auto-correlation series-wise connections along with trend and seasonal decomposition to achieve state-of-the-art time series forecasting performance [185].

We refer readers to Ref. [281] for further discussion in benchmarking transformer architectures for long-term learning in text sequence and Ref. [185] for review of transformers in time series forecasting. The studies show that no transformer is best for every task, and the performance varies across different tasks. Ref. [185] reviews transformers for time series forecasting applications and highlights their strengths and limitations through a taxonomy. Ref. [185] evaluates recent MTS transformers based on long sequence length and model size proportional learning analysis. Ref. [185] reveals the prediction accuracy of different transformer models rapidly deteriorates as input length and network depth increase—limiting the capability of many carefully constructed transformers for complex modeling in long-term forecasting tasks. The question of building a suitable and multi-layered transformer architecture that increases the model performance is still an open research challenge. Transformers do not make any assumptions about the characteristics and patterns of the data and need extensive data to prevent data from overfitting. Recent studies incorporate periodicity [179] or frequency processing [184] as inductive biases into a TS transformer to enhance performance significantly. Integration of GNN and transformer attentions have improved performance and enhanced understanding of the spatio-temporal dynamics and causality [129].

3.2.4 Graph Neural Networks

The CNNs are robust at capturing hidden patterns of structured Euclidean data, such as images, text, and regularly sampled time series data. Their effectiveness is limited in non-Euclidean domains—e.g., graphs with complex relationships and interdependencies between objects. In the realm of geometric deep learning (GDL), graph neural networks (GNN) generalize CNNs by extending structured deep neural network models to non-Euclidean domains [283]. Graphs consist of interconnected nodes—each with a set of features. GNNs become prevalent across a wide range of applications, such as computer vision [117–119,121], traffic monitoring [112], cybersecurity [114,115], environment monitoring [109], and the LHC [23,38,284,285].

Graph neural networks employ message-passing primitives that update the node representations or features by aggregating information passed from neighboring nodes [283]. Let a graph data $\mathcal{G}(\mathcal{V},\mathcal{E})$ is collection of nodes $\mathcal{V}$ (vertices) and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ between pairs of nodes. For nodes $\forall u \in \mathcal{V}$ with feature vector $\mathbf{x_u}$ and edge features $\mathbf{e_{uv}} : (u, v) \in \mathcal{E}$, the message passing neural networks on node u can be expressed as:

$$\mathbf{h_u} = \Phi\left(\mathbf{x_u}, \bigoplus_{v \in \mathcal{N}_u} \Psi(\mathbf{x_u}, \mathbf{x_v}, \mathbf{e_{uv}})\right) \tag{3.17}$$

where the $\mathcal{N}_u$ is the set of one-hop neighbors of the node u, and Φ and Ψ are the update and message functions of the network, respectively. The $\bigoplus$ is a permutation invariant aggregation operator that can accept an arbitrary number of inputs (e.g., element-wise sum, mean, or max). The $\mathbf{h_u} \in \mathbb{R}^M$ is the output feature vector of node u with the dimension of M.

Graph convolutional networks (GCN) are variants of GNNs that perform convolution-like operations on graphs [286,287]. A GCN layer defines a first-order approximation of a localized spectral filter on graphs. The convolution in GCN is the same as in CNNs, where the model learns the features by inspecting neighboring nodes. The major difference is that CNNs are built to operate on regular data, while GCNs are the generalized version of CNNs where the numbers and order of node connections vary [286,287]. Information propagates like a wave or signal across the nodes in spectral GCNs. A GCN layer involves three operators: graph convolution, linear layer, and nonlinear activation. It can be described from a message-passing perspective—for each node, 1) aggregate neighbors' representations to produce an intermediate representation, and 2) transform the aggregated representation with a linear projection followed by a non-linear activation [287]. The graph convolution operation produces a normalized sum of the node features of the neighbors on a given graph $\mathcal{G}(\mathcal{V},\mathcal{E})$:

$$\mathbf{h_u} = \sigma\left(\frac{1}{d_u} \sum_{v \in \mathcal{N}_u} \mathbf{W}\mathbf{x_v}\right) \tag{3.18}$$

where the $\mathcal{N}_u$ is the set of one-hop neighbors of the node u with self-looping to include u in the set. The d_u is the degree of the node, meaning the number of neighboring nodes $d_u = |\mathcal{N}_u|$. The d_u provides normalization to prevent numerical instabilities and vanishing or exploding gradients during model training. The lower the node degree, the stronger a node belongs to a certain group or cluster. The $\sigma(\cdot)$ denotes an activation function. The $\mathbf{h_u} \in \mathbb{R}^{1\times M}$ is the node feature vector with the dimension of M. The $\mathbf{W}$ is a shared weight matrix for node-wise feature transformation.

Ref. [287] proposes symmetric normalization using Laplacian matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ to improve the learning dynamics; the layer-wise propagation mechanism for a multi-layered GCN is formulated as:

$$H^{l+1} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^l\mathbf{W}^l\right) \tag{3.19}$$

where $H^l \in \mathbb{R}^{N\times M}$ is the matrix of node representation in the l^{th} layer; $H^0 = X$. $\mathcal{A}$ is the adjacency matrix of the graph $\mathcal{G}$, and $\mathbf{D} = diag\left(\sum_j \mathbf{A}_{ij}\right)$ is a degree matrix.

The graph attention networks (GAT) are another popular variant of GNN that incorporates an attention mechanism to capture the context of importance among the nodes [288]:

$$\mathbf{h_u} = \sigma\left(\frac{1}{d_u}\sum_{v\in\mathcal{N}_u} \alpha_{uv}\mathbf{W}\mathbf{x_v}\right) \tag{3.20}$$

where α_{uv} is the attention coefficient from node v to u.

Ref. [289] presents a gated graph sequence neural network (GGS-NN) that adopts GNNs for temporal data using RNNs for nodes with sequential features. The message passing of GGS-NN employs GRU for updating node features per each time step:

$$\mathbf{h_u}^{t+1} = \mathrm{GRU}\left(\mathbf{h_u}^t, \sum_{v\in\mathcal{N}_u} \mathbf{W}\mathbf{h_v}^t\right) \tag{3.21}$$

where $\mathbf{h}^t$ is node features at the t time step, respectively.

GNNs have achieved promising performance in MTS forecasting [175, 177] and AD [95, 108, 129]. There are specific challenges that need to be overcome when modeling multivariate data using GNNs: 1) the heterogeneity of sensors that monitor different quantities, and 2) more importantly, the graph edges—representing the relationships between sensors—are initially unknown in most settings. Refs. [95, 108, 177] utilize signal processing—linear correlation maps—to generate the GNN model input-output graph data. [129,177] propose graph edge learning during model training. Training deep GNNs can generally be tough. The node characteristics may converge to the same vector and become practically indistinguishable for deeper GNN; this phenomenon is called over-smoothing [290]. Another challenge is the over-squashing of information from many neighbors into fixed-size vectors. Researchers have suggested using edge-wise dropout, pairwise distance normalization, node-wise mean and variance normalization, and residual skip connections [291]. While these strategies allow training GNNs with shallow layers, further research is necessary for deep GNNs. An in-depth technical discussion of GNNs is available in Refs. [283, 292,293] and survey in Refs. [294,295]. Very recent work in Ref. [296] reviews GNNs for sequence molding in NLP.

RNNs and CNNs struggle to provide model decision and interaction explanations among multivariate data despite their dominance in TS modeling. We still rely on post-hoc model explanation tools—like Shapley additive explanations (SHAP) [221] and integrated gradient (IG) [85], and class activation map (CAM) [297]—to explain models in TS data. Despite the demonstrated promising efficacy of inferred post-hoc explanation in some TS data, a research gap exists to fully understand their effectiveness and development of a method that accounts for sequential characteristics [41, 297–300]. GNNs are emerging as promising approaches to explain relationships between variables without post-hoc methods and offer the potential to provide intuitive insights into complex data. Their network parameters—e.g., node attention scores—and generated graph networks—e.g., edge linkage and edge weight—equip interpretable relations among graph nodes [108]. Self-explainable models could benefit from future GNNs—particularly in influence prediction, causal interaction, and AD based on interaction network deviations.

Deep learning networks for TS modeling can greatly benefit from the long-sequence learning ability of RNNs and the parallel processing and attention capability of transformers. The multivariate interaction learning ability of GNNs can complement these capabilities and lead to the development of robust and interpretable models in future research [129, 185]. Hybrid models that incorporate different networks—RNNs, CNNs, transformers, and GNNs—tend to achieve better performance and are commonly utilized in real-world industrial applications [41, 70, 129, 186].

3.3 Architectures for Time Series Modeling

We will highlight below deep learning architectures for TS modeling—including feed-forward, deep autoencoders, deep variational autoencoders, generative adversarial networks, and sequence-to-sequence modeling.

3.3.1 Feed-Forward Architecture

Feature extraction and decision-making are the two main network blocks of a feed-forward architecture. The feature extraction block extracts important features from the input data, and the decision block maps the extracted features to the final output. This architecture is widely utilized for classification, regression, and unit-timestep forecasting tasks on various data types—including time series, texts, images, and graphs. The feature extraction network handles input data characteristics, and the decision network is typically fully-connected neural networks (FC) [297, 300, 301]; RNNs and 1DCNNs are widely employed as feature extraction networks to provide time invariant feature sets for FC classification decision networks [297, 301]. The accuracy of feature extraction plays a crucial role and often requires extensive training. Well-learned features encourage complexity simplification of the final decision networks. The two-network architecture also enables transfer learning in that pre-trained feature extraction networks—trained on different data or tasks—are utilized while fine-tuning only the decision networks on a given target data. The feed-forward architecture is typical for supervised learning approaches, whereas the encoder-decoder architecture is for unsupervised modeling. These architectures are often discussed in separate categories since the design concept differs.

3.3.2 Deep Autoencoders

An autoencoder (AE) is an unsupervised learning architecture consisting of encoder and decoder networks trained to reconstruct the input data. The encoder (E_θ) maps the input X to a latent space representation $\mathbf{z}$. The decoder (D_ω) then tries to reconstruct the input data $\bar{\mathrm{X}}$ from $\mathbf{z}$. The AE function $\mathcal{F}_{ae}$ can be formulated as:

$$\begin{aligned} \bar{\mathrm{X}} &= \mathcal{F}_{ae}(\mathrm{X}) \\ \mathbf{z} &= E_\theta(\mathrm{X}) \\ \bar{\mathrm{X}} &= D_\omega(\mathbf{z}) \end{aligned} \tag{3.22}$$

where the θ and ω are trainable parameters for the encoder and decoder networks, respectively. The networks commonly use RNNs, CNNs, and transformers to learn temporal characteristics. AEs are trained by optimizing a reconstruction loss function $\mathcal{L}(\mathrm{X}, \bar{\mathrm{X}})$. Although there are several types of loss functions, *mean absolute error (MAE)* and *mean squared error (MSE)* are among the widely employed.

$$\begin{aligned} \mathcal{L}_{mae} &= \mathbb{E}[|\mathrm{X} - \bar{\mathrm{X}}|] \\ \mathcal{L}_{mse} &= \mathbb{E}[(\mathrm{X} - \bar{\mathrm{X}})^2] \end{aligned} \tag{3.23}$$

where $\mathbb{E}$ is an expectation operator.

Deep autoencoders are AEs with deep layers in their encoder and decoder networks (see Fig. 3.4). They are potent tools widely found in feature extraction, dimensionality reduction, and data compression applications. Deep AEs often employ constrained—bottleneck—encoders to extract low dimensional latent features $\mathbf{z}$ from the high dimensional input data X. Deep AEs offer robust function generalization when utilizing nonlinear E_θ and D_ω networks, but they are also susceptible to overfitting as the number of model parameters increases—necessitating careful network design.

Deep AEs have been successfully employed for unsupervised AD modeling—trained to effectively reconstruct only healthy patterns so that the reconstruction error grows when encountering anomalous patterns. AD using AEs can be sensitive to model complexity and may not always guarantee higher reconstruction errors for anomalous patterns, as anomalous data may also be well reconstructed at times—leading to the omission of anomalies [41, 132]. Limited model complexity would hinder learning relevant patterns—resulting in higher reconstruction errors even

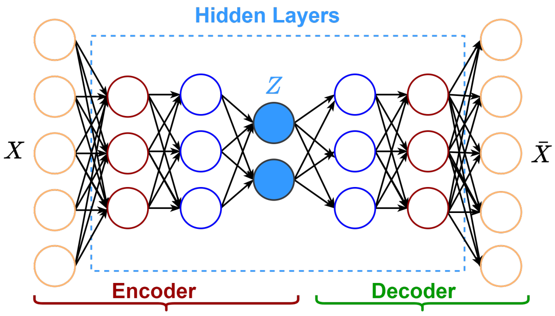


Figure 3.4: Deep autoencoder model. The X and X̄ are the input and reconstructed multivariate data, respectively.

for healthy patterns. Ref. [302] introduces the latent-insensitive autoencoder (LIS-AE)—employs unlabeled data from a similar domain as negative examples to shape the latent layer of the AE—to avoid reconstruction of anomalous patterns.

Anomaly detection can also be carried out on the latent space of an AE; it assumes dissimilarity in the latent characteristics of healthy and anomaly patterns [41, 303–305]. Algorithms based on such as density, probability, and distance estimation are employed to measure or model the discrepancy in latent values for AD [12]. These algorithms evaluate the difference between latent vectors of the input data and healthy reference data. The density-based approach identifies anomalies in data regions with low density or high variability. Probability- and distance-based methods compare the closeness of latent vectors of input data compared to the healthy data to detect anomalies. Since AEs may not entirely guarantee a variance in the latent space, and overlapping latent regions are not uncommon, they require careful design and model tweaking [302,304]. Employing both the reconstruction and latent AD may thus leverage the performance of AD since hyperparameter tuning of AEs heavily relies on empirical analysis [41, 303, 304].

3.3.3 Deep Variational Autoencoders

Variational autoencoders (VAEs) are probabilistic generative models with different mathematical formulations from regular AEs (see Fig. 3.5). The reconstruction decoder network becomes a data generator by taking input from random samples on the latent space [83]. VAEs provide control over the latent space distribution of the encoder to render continuous probabilistic latent representation that resembles a known distribution—commonly a Gaussian distribution $\mathcal{N}(\mu, \sigma)$.

The encoder and decoder model probabilistic conditional distributions in which the encoder (E_θ) approximates the posterior distribution $q_\theta(z \mid x)$ and the generative decoder (D_ω) models the prior likelihood distribution $p_\omega(\bar{x} \mid z)$, where θ and ω denote network parameters. The θ minimizes the distribution disparity between $q_\theta(z \mid x)$ and $p_\omega(z \mid x)$ during VAE model training, and the ω reduces the reconstruction error between the input x and the output $\bar{x}$. Kullback-Leibler divergence (KL) distance (D_{KL}) is frequently used to measure the difference between the two distributions.

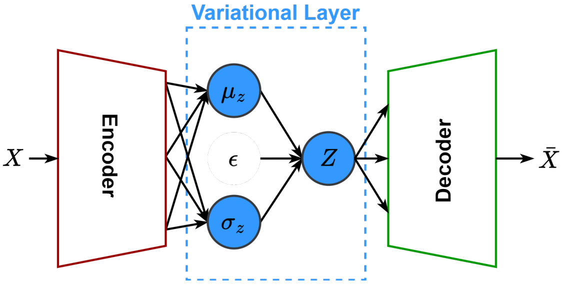


Figure 3.5: Deep variational autoencoder model. The X and X̄ are the input and reconstructed multivariate data, respectively. The $z = \mu_z + \sigma_z \odot \epsilon$, where the μ_z and σ_z are outputs of the encoder, the $\epsilon \ \sim \mathcal{N}(0, I)$, and $\odot$ signify an element-wise product.

The training loss function $\mathcal{L}_{vae}$ combines the two losses as:

$$\begin{aligned} \mathcal{L}_{KL} &= D_{KL}\left[q_\theta(z \mid x), p_\omega(z \mid x)\right] = \mathbb{E}_{z \sim q_\theta(.|x)}\left[ln \frac{q_\theta(z \mid x)}{p_\omega(z \mid x)}\right] \\ \mathcal{L}_{vae} &= \mathcal{L}_{rec} + \beta \mathcal{L}_{KL} \end{aligned} \tag{3.24}$$

where $\mathcal{L}_{rec}$ and $\mathcal{L}_{KL}$ are the reconstruction and regularizing KL loss, respectively. The β is a hyperparameter weight factor of the KL regularization. When training a VAE with the backpropagation, one cannot use the expectation operator $\mathbb{E}$ of the $\mathcal{L}_{KL}$ within the gradient operator. A reparameterization trick is employed to overcome this problem for $D_{KL}\left[\mathcal{N}(\mu_z, \sigma_z), \mathcal{N}(0, \mathbf{I})\right]$, that involves a standard random number generator $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $z = \mu_z + \sigma \odot \epsilon$. The $\odot$ signify an element-wise product [83].

Deep VAEs have also achieved promise in AD [41, 97, 125, 306, 307] besides their typical application in synthetic data generation [83]. The contribution of the VAEs for AD is relative robustness against overfitting in reconstruction and the probabilistic latent representation [41, 307]. The deviation in the KL loss can also provide an additional metric for detection [307].

3.3.4 Generative Adversarial Networks

Generative adversarial networks (GANs) are another type of generative modeling approach that is more robust than VAEs [308]. GANs utilize an adversarial training process that includes two networks: a generative model G_θ, and a discriminative model D_ω. The G_θ captures the data distribution and generates data that is close to the real training data, while the D_ω estimates the likelihood that a sample came from the training data instead of G_θ. The G_θ competes with the D_ω and the feedback from D_ω improves its accuracy. The improvement on G_θ aids the D_ω to enhance its discrimination sensitivity between real and fake data. The model training adversarial loss function is given as:

$$\min_{G_\theta} \max_{D_\omega} V(D_\omega, G_\theta) = \mathbb{E}_{x \sim P_x(x)}[\log D_\omega(x)] + \mathbb{E}_{z \sim P_z(z)}[\log(1 - D_\omega(G_\theta(z)))] \tag{3.25}$$

Training GAN has several failure modes and still poses great theoretical and empirical challenges [309–311]. Training through first-order optimization methods such as gradient descent commonly fails to converge to a stable solution where the G_θ and D_ω cannot improve their objective [312, 313]. Another challenge of GANs is mode collapse failing to generalize properly—missing entire modes from the input data [311]. This may happen when the generator learns quickly compared to the D_ω, a weak discriminative network that fails to notice the pattern of omission or bad choice of an objective function [310]. GANs may suffer from a vanishing gradient problem when the D_ω learns quickly and becomes perfect before the generator; it can almost perfectly distinguish real and fake data—leaving the G_θ stuck in a very high loss with a very small gradient to pursue the right direction. Ref. [311] presents the Wasserstein GAN by modifying the discriminator and the loss function to mitigate mode collapse and vanishing gradients issues. Ref. [312] proposes a progressive curriculum learning scheme that starts with simple generation and scales up to achieve convergence on GAN. Ref. [313] discusses stable convergence GAN using a two-time scale update rule (TTUR) method that employs a lower learning rate for the generator to slowly drive the discriminator while capturing the required generation information. Building effective networks for the G_θ and D_ω and formulating optimal training loss functions are active research areas in GANs [314].

The G and D are made of sequential data handling networks such as the RNNs [131, 315–318] and CNNs [134] for time series modeling. GANs have been studied primarily in TS generation [315–318] and AD applications [131, 134]. AD using GANs differs from AE-based AD mechanisms and can be more complicated. The discriminator can be used directly to detect anomalies; the generator complicates the AD since GANs do not allow direct estimation of latent variables from input data [131, 134].

3.3.5 Sequence-to-Sequence

Sequence-to-sequence (S2S) is an encoder-decoder architecture and is essentially similar to TS AEs—both first extract representational context features from the input sequence data and construct the target sequence data from the context features [256]. An encoder-decoder architecture can be employed for input data reconstruction—like AEs—or prediction of different target variables with different time and feature dimensions—like S2S. The sequential generation of S2S is also slightly different: the decoder uses the context vector of the encoder and the hidden state of the decoder along with the previously generated token to iteratively generate the output sequence—one or a sequence of tokens at a time [255, 256]. Sequence-to-sequence modeling is widely applied to sequence prediction in language translation [255, 256], text summarization [319, 320], conversational models [319], image captioning [321], and multi-time step forecasting [70, 167, 179, 282]. The recent large pre-trained language models (LLMs)—including Google's BERT [322], Amazon's AlexaTM [323], Meta's LLaMA [324] and OpenAI's ChatGPT [325]—utilizes S2S modeling.

Sequence-to-sequence models are known for their flexibility in handling a wide range of tasks and variable-length input and output sequences [70,167,179,255,256, 282,319,321]. One of the major challenges is their computational expense during training, which makes them difficult to optimize. S2S models may overfit the training data—leading to poor performance on new data—if not properly regularized. They may also face difficulty handling very long input sequences as the context vector may only capture some of the information in the input sequence—creating a bottleneck [70,255]. Attention mechanisms allow the network to focus on the input sequence selectively and overcome the bottleneck—making it easier to handle longer sequences [70,167,255]. Attention improves the encoder-decoder architecture performance by allowing the model to automatically search for relevant parts of a source sequence that are relevant for predicting the target output [255]. RNNs with attention mechanisms dominate the S2S architecture arena [70,255,319,320], and recent studies on NLP tasks are also engaging transformers for their parallel processing, multi-head attention, and faster training [167,179,282,321].

3.4 Explainable AI for Time Series Modeling

Deep learning models are black-box models that are challenging to explain or interpret. This difficulty curtails the wider model deployment in many real-world applications; black-box systems lack transparency and can be more difficult to monitor and regulate. An explainable AI (XAI) aims to increase the confidence, stability, and robustness of AI models [326]. XAI investigates various objectives—including explainability, interpretability, trustworthiness, interactivity, stability, robustness, reproducibility, and confidence [326]. Explainability and interpretability are the two primary concepts examined to establish trustworthiness through XAI [326,326]. Although the definition of the XAI terms varies in the literature [327,328], we use the following definitions:

Definition 14. *Explainability* refers to the decision-making process explanation of models from the input to the output.

Definition 15. *Interpretability* refers to the passive characteristic of models and refers to the requirement of a meaningful or sensible explanation to humans.

XAI discusses post-hoc and ante-hoc methods in the literature for DL models [327,329]. Post-hoc methods—predominantly utilized—involve approximating ML model behavior by identifying relationships between feature values and predictions. These methods can be model-agnostic or model-specific knowledge extraction approaches [327]. A vast effort has been devoted to developing post-hoc methods to understand and explain black-box DL models [221,330,331]. Ante-hoc methods integrate explainability into the ML model architecture to create a fully self-explainable model (glass-box) [329]. [329] strongly argues against black-box ML models for high-stakes decisions in criminal justice, healthcare, and computer vision and advocates for interpretable models instead to avoid bias and promote fairness.

Researchers, developers, and policymakers are still debating the effectiveness of post-hoc and ante-hoc approaches [329, 332]. There is also an ongoing discussion against explainability in relation to model complexity [333]. [333] argues that having capable and complex models is more advantageous than a functionalist understanding of models, while others argue the opposite [332, 333]. XAI is a broad concept with diverse techniques in the literature, and we will limit our discussion below only to selected studies on XAI for TS modeling.

The area of explainability for TS modeling has remained relatively unexplored despite the advancements in XAI [326, 331, 334]. The XAI for multivariate TS is significantly challenging—less intuitive than other data types like images [326, 335]. A few studies thus far have presented XAI approaches for TS modeling [326, 331, 334]. Refs. [326, 331] review XAI techniques applied to time series data. The authors discuss the types of generated explanations and the limitations and point out potential research directions. Ref. [15] presents elaborated measuring methods to quantify explanations for MTS AD models.

Refs. [297,300] employ a class activation map (CAM)—enabled by global average pooling in a CNN model—to discover contributing regions in the raw TS data for a TS classification (TSC) task. GradCAM extends the capability of CAM into other network types—beyond global pooling—using gradients to compute the CAM [299, 300]. Ref. [334] discusses visualizations of the most influential saliency input points for a particular prediction through gradients for univariate TSC models. [41, 336] demonstrate the feasibility of post-hoc explanations of Shapley additive explanations (SHAP) [221] and integrated gradient (IG) [85] for TS AD explanation. The lack of AD data sets with adequate causality annotation limits the research in XAI on TS data [15]. TSViz [335] presents a visualization framework for demystifying convolutional filters of DL models in TS analysis. They propose a visual saliency explanation and clustering of the convolutional filters for determining the primary sources of variation learned by the network through inverse optimization. They have identified parts of the network responding to a particular stimulus with the computed influences. The accuracy of the models degraded when filters with high influence scores were pruned or slight perturbation was added at the saliency regions of the input data. N-BEATS [337] builds an interpretable DL architecture with backward and forward residual links—provide interpretable results by decomposing the time series data into trend and seasonality on a univariate forecasting model.

Ref. [180] proposes the integration of a DL with state-space modeling (SSM) for explainable TS modeling. The concept combines the learning complexity benefit of DL with the easy interpretability of SSM. A transformer learns the temporal patterns and directly estimates the parameters of the SSM; the SSM provides interpretability by decomposing the time series data into trend and seasonality. The attention of the transformer is also utilized to identify which parts of the past history are most important for the prediction. This study did not provide interpretability from the interaction of multivariate variables. We will discuss below the background concepts of post-hoc explanation methods since we employ the methods to explain the prediction of our black-box AD models.

3.4.1 Explanation using Feature Attribution

Post-hoc XAI methods are dominantly employed to explain the predictions of ML models and facilitate model deployment into real-world settings. There are two main post-hoc model explanation methods categories: prediction-level and dataset-level explanations—also known as local and global interpretations [327]. Prediction-level approaches describe individual predictions made by models—the contribution of features or interactions that led to the specific prediction [336]. Dataset-level approaches focus on capturing the global relationships in the model—e.g., the global feature importance rank by measuring prediction accuracy degradation when the input variables exhibit randomly permuted values [338, 339]. Prediction-level insights provide detailed information on individual predictions, but explanations from specific predictions do not necessarily represent the global explanation of the model. AD applications can benefit more from local interpretability since different anomalies may require different explanations [41].

Several variants of prediction-level feature attribution explanation algorithms—including LIME [330], SHAP [221], DeepLIFT [340], IG [85], SmoothGrad [341], and GradCAM [299, 300]—have been proposed for deep learning models. The state-of-the-art prediction-level explanation approaches generally follow perturbation [221, 330] and gradient [85, 299, 341] techniques. Perturbation-based methods, such as LIME [330], and SHAP [221], generate the explanation by evaluating the output change after replacing the features with randomly perturbed variables [221, 330]. They explain an output prediction of a given ML model by approximating it locally with an interpretable linear model around the prediction [221,330]. Gradient-based methods, such as IG [85] and SmoothGrad [341], explain predictions by tracking the backpropagation gradients of the outputs with respect to input instances [85, 299, 341]. IG variants are axiomatic explanation algorithms that compute the relevance score to input features by approximating the integral of gradients—sensitivity. In our study in Ref. [41], we combine SHAP [221] and IG [85] to capture the importance of the MTS input sensors on the detected anomalies.

SHapley Additive exPlanations (SHAP): The SHAP methods explain individual predictions based on game theory for optimal Shapley values [221]. The Shapley values show how to distribute the prediction among the features fairly, and the explanation is represented as an additive feature attribution method:

$$\mathcal{F}(\mathrm{X}) \approx \mathcal{G}\left(\mathrm{X}'\right) = \phi_0 + \sum_{\forall i} \phi_i x'_i \tag{3.26}$$

where the $\mathcal{F}$ and $\mathcal{G}$ are the original prediction and the explanation linear models, respectively. The $x'_j \in \{0,1\}^M$ is simplified input features (1 denotes feature presence in the coalition, and 0 otherwise), and M is the number of simplified input features. The $\phi_j \in \mathbb{R}$ is the Shapley feature attribution value for an input feature x_i, and ϕ_0 represents the model output with all simplified inputs missing. The ϕ_i is estimated to meet the desirable properties of SHAP—local accuracy, missingness, and consistency:

$$\phi_i = \sum_{S \in N \setminus x_i} \frac{|S|!(M - |S| - 1)!}{M!} \left[f_x(S \cup \{x_i\}) - f_x(S) \right] \tag{3.27}$$

where S is the set of non-zero indexes in x', and N is the set of all input features.

There are several variants of SHAP proposed in the literature, such as a model-agnostic KernelSHAP, TreeSHAP for tree-based models, and DeepSHAP for deep neural networks. The KernelSHAP is computationally very slow and impractical to compute Shapley values for many instances. The TreeSHAP is an efficient estimation version of SHAP for tree-based machine learning models [84]. It defines the value function using the conditional expectation instead of the marginal expectation—KernelSHAP—which ignores feature dependence like most permutation-based methods. TreeSHAP reduces the computational complexity of computing exact KernelSHAP values from $O\left(TL2^M\right)$ to $O\left(TLD^2\right)$ by approximation for tree-based models, where T is the number of trees, L is the maximum number of leaves in the trees, and D is the maximum depth of any tree. The limitation of the conditional expectation of TreeSHAP is that features that do not influence the prediction function can get a TreeSHAP estimate different from zero; a non-zero estimate can happen when the feature is correlated with another feature that actually influences the prediction. The DeepSHAP combines the DeepLIFT [340] method with the SHAP [221] framework for deep neural networks to improve ML model explanation. Ref. [342] recently introduces baseline Shapley (BShap) for global interpretation and local explanation of machine learning algorithms.

Integrated Gradients (IG): The IG is another local attribution method based on an axiomatic model explanation algorithm—using sensitivity and implementation invariance axioms [85, 341]. IG is also based on the Shapely concept but with a generalization to include continuous settings [342]. It computes the relevance score to each input feature by approximating the integral of gradients—sensitivity—of the DL network model's output with respect to the inputs along the path from given baselines or references to inputs [85]. IG assumes that a given two networks are functionally equivalent—invariance—when their outputs are equal for all inputs or have identical attributions despite having very different implementations. Given a target input X and a network function $\mathcal{F}$, feature attribution methods assign an importance score $\phi_i(\mathcal{F}, \mathrm{X})$ to the i^{th} feature value representing how much that feature adds or subtracts from the network output:

$$\phi_i^{IG}\left(\mathcal{F}, \mathrm{X}, \mathrm{X}'\right) = \overbrace{\left(x_i - x_i'\right)}^{\text{Difference from baseline}} \times \underbrace{\int_{\alpha=0}^{1}}_{\text{From baseline to input}} \overbrace{\frac{\delta \mathcal{F}\left(\mathrm{X}' + \alpha\left(\mathrm{X} - \mathrm{X}'\right)\right)}{\delta x_i} d\alpha}^{\text{accumulate local gradients}} \tag{3.28}$$

where X is the current input, $\mathcal{F}$ is the model function, and X' is some baseline input that represents the "absence" of feature input. The subscript i denotes indexing into the i^{th} feature. A large positive or negative ϕ_i indicates that the feature strongly increases or decreases the network output, respectively; an importance score close to zero indicates that the feature in question did not influence $\mathcal{F}$. The IG method is applicable only when the gradient of the prediction score with respect to the base features is defined—deep learning networks; it does not apply to models like tree ensembles since their response surface is not differentiable [342].

3.4.2 Quantifying Explainability and Interpretability

The consensus on evaluation mechanisms for explanation and interpretation algorithms is still developing and recent studies have started to address it [15, 307, 327, 328, 331, 343]. The studies discuss the desired properties of the XAI models [328], evaluating the provided explanation [331, 343], guidance for effective evaluation [327], and quantifying the evaluation based on the end target application [15, 307]. Ref. [327] presents the PDR framework with three criteria for selecting explanation methods for a given problem: predictive accuracy, descriptive accuracy, and relevancy. Predictive accuracy refers to the ability of a model to approximate the underlying data relationships. The descriptive accuracy measures the extent to which an explanation method can approximate what the model has learned. Relevancy deals with the importance of the extracted explanation for the target audience. Ref. [331] formulates sensitivity measuring metrics of post-hoc XAI solutions, such as LIME [330], IG [85], and SmoothGrad (SG) [341], for TS CNN models. Ref. [331] proposes several metrics—including max short-term sensitivity (MSS), max long-term sensitivity (MLS), average short-term sensitivity (ASS), and average long-term sensitivity (ALS)—to evaluate the sensitivity of the XAI models on a synthetic and real TS data sets. The study indicates higher sensitivity of the post-hoc methods for a smaller time window data, and changes in the network parameters and data properties fluctuate their response. A recent study in [15] introduces the Exathlon approach to explain collective anomalies—occurring over a period of time—in multivariate TS data. The authors propose four evaluation criteria to quantify the results of anomaly detection: anomaly existence, range detection, early detection, and exactly-once detection (see Fig. 3.6). Anomaly existence refers to flagging the presence of an anomaly somewhere within the anomaly interval, which consists of the root cause interval (RCI) and the extended effect interval (EEI). Range detection measures the precise time range of an anomaly. Early detection evaluates the latency between flagging the anomaly and the start time of the corresponding RCI. Exactly-once detection measures the ability of the algorithm to report each anomaly instance exactly once and avoid duplicate alerts. The authors use customizable range-based accuracy evaluation metrics to assess how well an AD algorithm can meet these four functionality levels.

Ref. [307] adopts a *HitRate@P* metric—from recommender systems in Ref. [344]—to quantify the interpretation using the anomaly flag hit counts from each TS variable for their OmniAnomaly AD model—employs GRU and non-Gaussian-VAE. The AD model generates the contribution from all variables into a list AS_t—

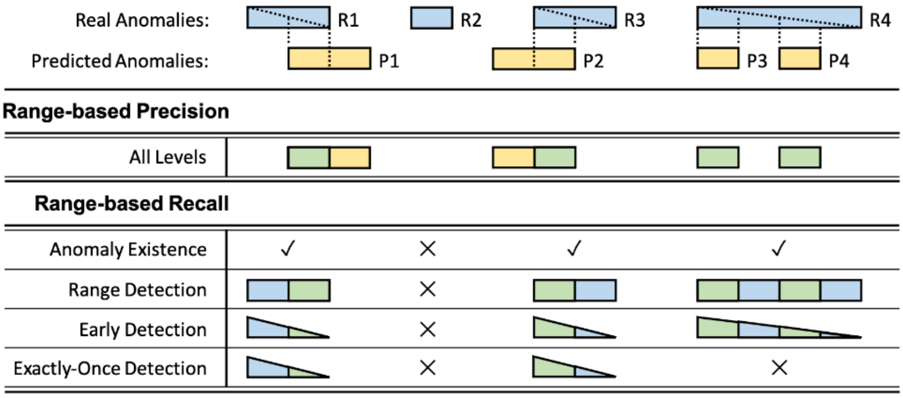


Figure 3.6: The Exathlon evaluation criteria for MTS AD [15]. The precision evaluates prediction quality (green out of yellow for each Pi). The recall evaluates anomaly coverage (green out of blue for each Ri).

ordered by their contributions—for a detected anomaly $\mathbf{a_t}$. Let GT_t be the ground truth list containing the univariate variables indeed contributing to anomaly $\mathbf{a_t}$, and the $HitRate@P$ is given as:

$$HitRate@P\% = \frac{\text{Hit } @top \lfloor P\% \times |GT_t| \rfloor}{|GT_t|} \tag{3.29}$$

where $|GT_t|$ is the length of GT_t and P set 100 or 150 in Ref. [344]. $Hit@P\%$ equals the number of overlapping dimensions between ground truth GT_t and the $top\lfloor P\% \times |GT_t|\rfloor$ contributing dimensions in AS_t suggested by an AD model. Lets use a toy example to explain $HitRate@P\%$ on a 4-variable observations with $AS_t = \{2, 1, 3, 4\}$ and $GT_t = \{1, 3\}$. This results in $|GT_t| = 2$, and $top\lfloor 100\% \times |GT_t|\rfloor = 1$, as only one matched—$\{1\}$—with GT_t from the top two of $AS_t \rightarrow \{2, 1\}$; the $HitRate@100\%$ becomes 0.5. Similarly, $HitRate@150\% = 1.0$, as the $top\lfloor 150\% \times |GT_t|\rfloor = 2$ (two matched, $\{1, 3\}$, from the top three, $150\%|GT_t| = 3$, of $AS_t \rightarrow \{2, 1, 3\}$).

The Exathlon [15] extends the explanation formulation approach of Ref. [307] for quantifying MTS AD results. The authors follow a similar approach of $HitRate@P$ [307, 344] for multivariate consideration and add more desired properties for local and global explanations, such as conciseness, consistency, and accuracy. The conciseness property corresponds to the number of features used in the explanation. Consistency refers to anomalies of the same type—that occur in a similar context—are expected to have consistent stability explanations. The accuracy properties measure the accuracy of the explanation on similar anomaly instances using a predictive model. Exathlon employs hit matching algorithms between GT_t and AS_t to quantify conciseness. It utilizes a count metric of how many variables in the MTS with anomaly reports have actual anomalies to measure conciseness. A subsampling procedure is used from the same anomaly time range to measure consistency—generating several subsamples and expecting the anomaly explanation to remain the same.

It is important to measure the explanations generated by explainable AI to ensure that the predictions made by deep networks are trustworthy. The measurement of these explanations can also help distinguish a more robust model from a weaker one, even when both models make identical predictions [299, 307]. Future research on graph neural networks and attention mechanisms has the potential to enable explanations in quasi-anti-hoc mode while exploiting the power of deep learning.

3.5 Impact of Data Normalization

Deep learning models involve multi-layer architecture involving several networks and activation functions. Scaling the modeling data sets to the operating ranges of the models is important to enhance training—mitigate saturation, and improve regularization [345–347]. Multivariate data can have a different range of values, and data normalization reduces the scaling heterogeneity that enhances learning and speeds up convergence. There are several normalization techniques—including but not limited to:

- *Min-max*: $\mathbf{x}^s = (\mathbf{x} - \mathbf{x}_{\min})(s_{\max} - s_{\min})/(\mathbf{x}_{\max} - \mathbf{x}_{\min}) + s_{\min}$. Maps $\mathbf{x} \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}] \to \mathbf{x}^s \in [s_{\min}, s_{\max}]$.
- *Z-scale*: $\mathbf{x}^s = (\mathbf{x} - \mu_{\mathbf{x}})/\sigma_{\mathbf{x}}$. Maps $\mathbf{x} \in \mathcal{D}(\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}) \to \mathbf{x}^s \in \mathcal{D}(0, 1)$.
- *Decimal scaling*: $\mathbf{x}^s = \mathbf{x}/K$. Maps $\mathbf{x} \in [\mathbf{x}_{\min}, \mathbf{x}_{max}] \to \mathbf{x}^s \in [\mathbf{x}_{\min}/K, \mathbf{x}_{\max}/K]$.
- *Sigmoid*: $\mathbf{x}^s = 1/(1 + e^{-\mathbf{x}})$. Maps $\mathbf{x} \in [\mathbf{x}_{\min}, \mathbf{x}_{max}] \to \mathbf{x}^s \in [0, 1]$.
- *Tanh*: $\mathbf{x}^s = (e^{\mathbf{x}} - e^{-\mathbf{x}})/(e^{\mathbf{x}} + e^{-\mathbf{x}})$. Maps $\mathbf{x} \in [\mathbf{x}_{\min}, \mathbf{x}_{max}] \to \mathbf{x}^s \in [-1, 1]$.

where $\mathbf{x}$ and $\mathbf{x}^s$ are the raw and scaled data, respectively. The subscripts *min*, *max*, and superscripts μ, σ denote the minimum, maximum, mean, and standard deviation values, respectively. $\mathcal{D}(\mu, \sigma)$ designates data distribution function with mean μ and deviation σ.

The choice of normalization techniques can significantly affect the prediction performance. Most DL models for TSAD have reported a low efficacy or exhibited high sensitivity to hyperparameters when evaluated on new sittings [12,81,130]. The lack of a robust generic normalization approach could be the reason for the reported performance deterioration in several of the models. Autoformer [179] employs a seasonal-trend decomposition for normalization for time series forecasting. It has utilized a moving average trend decomposition to achieve a considerable accuracy increase—by around 40%. Ref. [185] highlights the importance of normalization in deep learning for time series modeling. The authors applied the seasonal-trend decomposition from Ref. [179] to various recent time series forecasting transformers, such as Autoformer [179], Fedformer [184], Informer [167], TFT [178], and SSD-NET [180]. The results indicate that normalization contributes significantly to the performance boost of 50% to 80%—significantly higher gain than the difference in model architecture.

Machine learning models often encounter out-of-the-training data during model inference—a data concept drift phenomenon that occurs when the training data do not adequately cover the statistical distribution of the data [41, 348]. Models may struggle to keep accuracy even when trained with normalized data sets in such situations. Ref. [242] suggests a simple initialization scheme for TCN weights-CNN kernel weights initialized to generate average values—to alleviate the challenge and avoid normalization. They also propose a normalized absolute deviation loss function—total reconstruction error divided by the total target value. We found the approach promising, but it might be challenging to apply the initialization scheme to complex multilayer FC and RNN networks. Ref. [348] propose a simple reversible instance normalization (RIN) method that can effectively improve the performance of time-series forecasting models in the presence of out-of-range data distribution shifts. The approach involves standardizing the data input per time window slice and then applying a reverse operation to generate the final output. The RIN has demonstrated significant performance progress across several models. Out-of-range data normalization is necessary for various time series modeling problems beyond forecasting tasks, and different modeling tasks may require specific normalization techniques to achieve the desired outcome.

Chapter 4

Anomaly Detection

This chapter presents our studies on deep learning modeling for multivariate anomaly detection on infrastructure monitoring and data quality monitoring.

4.1 Unsupervised Deep Variational Model for Multivariate Sensor Anomaly Detection

The ever-increasing complexity of the CMS detector triggers a call for increasing automation since the quality of collected physics data hinges on the healthy operation of the detector at the time of data-taking. We present a data-driven unsupervised anomaly detection using a deep learning model (CGVAE) to rapidly identify detector system anomalies from multivariate time series sensor data of the HCAL. The CGVAE model consists of a variational autoencoder with convolutional and gated recurrent unit networks for fast localized feature extraction, long temporal characteristics capturing, and descriptive representation learning. The CGVAE employs encoded latent and reconstruction metrics for anomaly detection to mitigate signal reconstruction overfitting on anomalous patterns. The model integrates feature attribution algorithms to explain the contribution of the input sensors to the detected anomalies. The experimental evaluation on large sensor data sets of the HCAL demonstrates the efficacy of the proposed model in capturing temporal anomalies.

Several studies have proposed ML models for AD applications for the LHC systems [52,56] and DQM [39,40,50,52,56,67,127]. There have been few efforts carried out thus far to exploit temporal sensor data despite the acknowledged potential in the future CERN automation technology challenges [39]. Refs. [52,56] have explored time-series deep learning for safety monitoring of the LHC accelerator magnetic systems. There are limited tangible efforts in exploring ML algorithms to underpin system monitoring automation of the CMS detector through AD models [69]. This study focuses on developing a time-aware, unsupervised AD model for HCAL from multidimensional readout boxes (RBXes) monitoring signals. The multivariate time series sensor data includes readings from voltage, current, and power sensors of various components of the ngCCM of the HCAL.

Although abundant studies have presented deep AD models using different combinations of convolution network, recurrent network, and variational autoencoder for time series data [95–97, 125], our proposed model is the first to combine all the three blocks for fast localized multidimensional feature extraction, long temporal behavior capturing, and probabilistic representational learning, respectively. Previous studies on AE-based temporal AD models [95–98, 125] rely on AD metrics either from reconstruction error or latent features. AE-based AD models may struggle to generate high reconstruction errors or distinguishable latent mappings for anomalous signals [303]. We propose employing reconstruction and latent attributes to mitigate over-fitting limitations on anomalous data.

We validate the efficacy of the proposed AD model in detecting temporal point anomalies (spikes and dips) and discords (unusual time series subsequences) on 34 RBXes. We compare the AD performance with alerts generated from the ngCCM's error counter quantities—current fault monitoring variables at the HCAL. We incorporate an ablation study on the CGVAE model to demonstrate the relevance and contribution of the model blocks. We integrate feature attribution algorithms into our AD system to capture potential influence from the monitored sensors to the generated anomaly flags. We demonstrate that feature attributions can be combined to provide end-to-end output explanation when a given system consists of more than one model—deep autoencoder and novelty detector based on isolation forest in our AD system. The key contributions of our work are highlighted below:

1. We present the first study on deep learning for time-aware AD for RBXes monitoring from diagnostic sensor data.

2. We introduce CGVAE to detect multivariate temporally contextual anomalies exploiting both reconstruction errors and novelty scores on encoded latent features along with detection explanation—using pipelined feature attribution methods.

4.1.1 Dataset Description

There are two distinct sets of data that come from the front-end electronics of the HCAL: 1) *physics data sets*, which contain all the measurements used to reconstruct the properties of particles detected by the HCAL, and 2) *electronics sensor data sets*, which are recorded for detector health monitoring and diagnostic purposes, and do not contain any measurements that are used for physics data analysis. These diagnostic sensor data are recorded whenever the electronics are powered on—regardless of whether particles enter the calorimeter. Our study focuses on developing an AD model on those diagnostic sensor data and does not use any physics data sets.

We utilized the ngCCM sensor data from the HE detector collected in 2018 using the ngCCM server. The ngCCM server is software that handles the communications of the front-end controller interface for accessing the ngCCM. The data set contains one-minute 420M samples from around 3.4K monitored quantities of 34 active RBXes (HEP01-18 and HEM01-18, excluding HEM15 and HEM16) from September 15 to December 10, 2018. From 100 monitored sensors for each RBX, 69 relevant monitoring quantities are identified and categorized into 32 sensors with continuous values, 32 counters with monotonically increasing discrete values, and 5 flag variables with categorical values—after quantities with constant values and large missing periods are removed. We selected only twenty-eight sensors for the AD modeling and certain counters for model validation. The sensor signals are composed of current, voltage, and optical power measurements of various components of the ngCCM. We downsampled the data into 10-minute intervals by averaging to capture the relevant temporal information, as most of the sensors remain constant for smaller time intervals.

4.1.2 Methodology

The requirement of the AD model for complex systems of the HE RBXes poses certain challenges: e.g., capturing system-level anomalies—from the interaction of multivariate IS signals—and sensor-level anomalies, scalability to all RBXes, generating low-dimensional representations, and explainable AD. We employ a variational autoencoder (VAE), which utilizes latent and reconstruction anomaly scores for robust AD and generates low-dimensional system representation. We propose a single, unsupervised, and end-to-end deep AD model for monitoring all RBXes through learning generalization—employed adaptive data preprocessing and training data incorporation from multiple RBXes to achieve scalability.

We employ a deep VAE model for the AD system. The VAE $\mathcal{F}$ operates on the MTS senosr data $\mathrm{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_x}] \in \mathbb{R}^{T \times N_x}$—a multivariate sequence with N_x UTS sensor variables in a time window $[t-T, t]$. The $\mathcal{F}_{\theta,\omega} : \mathrm{X} \rightarrow \bar{\mathrm{X}}$, parametrized by θ and ω, attempts to reconstruct the input X and outputs $\bar{\mathrm{X}}$ data. The encoder network of the model $E_\theta : \mathrm{X} \rightarrow \mathbf{z}$ provides low-dimensional latent space, $\mathbf{z} = E_\theta(\mathrm{X})$, and the decoder $D_\omega : \mathbf{z} \rightarrow \bar{\mathrm{X}}$, reconstructs the ST data from $\mathbf{z}$, $\bar{\mathrm{X}} = D_\omega(\mathbf{z})$ as:

$$\bar{\mathrm{X}} = \mathcal{F}_{\theta,\omega}(\mathrm{X}) = D_\omega(E_\theta(\mathrm{X})) \tag{4.1}$$

We present the CGVAE AD model that detects deviation- and proximity-based anomalies using reconstruction errors and novelty scores from the distribution of the encoded latent features, respectively (see Fig. 4.1). The AD model employs a constrained VAE with convolutional (CNN) and gated recurrent unit layers (GRU). We employ an isolation forest algorithm (IF) for novelty detection from the latent space of the AE; the encoder of the AE acts as a feature extraction block for the IF [349]. We propose a mechanism of aggregation of multiple feature attribution (FA) algorithms—*integrated gradients (IG)* [85] and *TreeSHAP* [84]—to provide output explanation for the detected latent anomalies from the input TS signals.

The proposed AD system is composed of two subsystems.

1. Sensor AD: detects anomalies for each UTS sensor $\mathbf{x}$ using reconstruction deviation as abnormality scores.

2. System AD: detects anomalies manifested in MTS sensors using novelty AD detection on the latent features and aggregate reconstruction anomaly scores from the sensor AD.

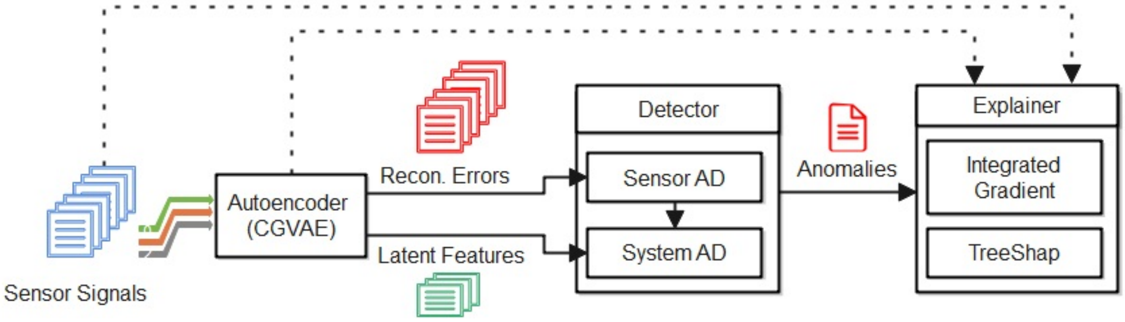


Figure 4.1: System design of our proposed AD system. The AE yields reconstruction scores and encodes latent features from the input signals. The detector performs anomaly detection decisions. The explainer yields the influence of the input MTS signals on the detected anomalies.

The AD system estimates various anomaly scores using a sliding time window T. Time windowed scores effectively capture sequence collective anomalies—discords—and strong point anomalies. The sensor AD uses reconstruction error $\mathbf{e}_i$ at time t based on mean absolute error (MAE) as:

$$\mathbf{e}_i(t) = \frac{1}{T} \sum_{t'=t-T}^{t} |\mathbf{x}_i(t') - \bar{\mathbf{x}}_i(t')| \tag{4.2}$$

where $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$ are the input and reconstructed univariate data for the s^{th} sensor. We generate the anomaly flags $\mathbf{A}_s = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N]$ after applying threshold $K_s = [k_1, k_2, \ldots, k_N]$ on the standardized anomaly scores $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N]$ as:

$$\begin{aligned} \mathbf{s}_i &= (\mathbf{e}_i - \mu_i)/\sigma_i \\ \mathbf{a}_i &= \mathbf{s}_i > k_i \end{aligned} \tag{4.3}$$

where $\mu_i = \mathbb{E}[e_i]$ and $\sigma_i = \sqrt{\mathbb{E}[(e_i - \mu_i)^2]}$ are the mean and standard deviation, respectively, calculated on the training data set. The reconstruction score is standardized to compensate for the reconstruction performance variations across the sensors. Each sensor has its own adjustable decision threshold k_i to enable sensitivity tuning of the AD without triggering the need for model retraining.

The system AD reports multivariate anomalies using latent AD through the IF model and multivariate reconstruction AD. The IF model is formulated as:

$$\begin{aligned} S_{LD} &= \mathcal{H}(\mathbf{z}) \\ A_{LD} &= S_{LD} > K_{LD} \end{aligned} \tag{4.4}$$

where the $\mathcal{H}$ is the IF model, and S_{LD} and A_{LD} are the latent outlier score and anomaly flag of the latent AD.

The multidimensional reconstruction scores S, and flags $A = [a_i, \ldots, a_N]$ from the sensor AD are aggregated using *Mahalanobis distance* (MD) estimation [350] and *additive flag scoring* (AFS) method [307], respectively. The MD is an effective multivariate distance metric between a point (vector) and is calculated as:

$$d_{MD} = \sqrt{(\mathbf{s}_j - \mathbf{s}_\mu)^T \cdot C^{-1} \cdot (\mathbf{s}_j - \mathbf{s}_\mu)} \tag{4.5}$$

where d_{MD} is the MD distance of the multivariate reconstruction error. The vector s_j is the multivariate anomaly score of the j^{th} observation, the vector $\mathbf{s}_\mu = \mu_1, \mu_2, \ldots, \mu_N$ contains the mean values of the univariate anomaly scores estimated on the training dataset, and C^{-1} is the inverse covariance matrix of S. We generate the MD anomaly flags through $A_{MD} = d_{MD} > K_{MD}\mu_{MD}$, where K_{MD} is the detection sensitivity, and $\mu_{MD} = \mathbb{E}[d_{MD}]$ is the mean distance of the healthy data and calculated on the training dataset.

The AFS is a simple method that sums up univariate anomaly flags $\mathbf{a}_i$ from each sensor as a system anomaly score:

$$\begin{gathered} S_{AFS} = \sum_{i=1}^{N} \mathbf{a}_i \\ A_{AFS} = S_{AFS} > K_{AFS} \end{gathered} \tag{4.6}$$

where S_{AFS} and A_{AFS} are the count score and anomaly flag of the AFS metric. We employ AFS to spot group and isolated anomalies across the multidimensional monitoring sensors.

The final system AD combines all the multivariate AD flags using a union aggregation:

$$A_{MTS} = A_{LD} \cup A_{MD} \cup A_{AFS} \tag{4.7}$$

Model Architecture and Training : The AE of the proposed CGVAE AD model combines the merits of 1DCNN[1] RNN [2] and VAE (see Fig. 4.2). We employ three layers of CNNs for fast and effective localized feature extraction from the sensor signals. The shared nature of the filters in the CNN allows for capturing similar features throughout the multivariate sequences, reducing the number of trainable parameters compared to fully-connected neural networks. We integrate two layers of GRU networks into the AE to capture long-term temporal dependencies from the extracted localized features. We chose GRU over the other RNN variants because of its simplified network with fewer trainable parameters but capable of achieving comparable performance [351]. We apply VAE [83] to enforce continuous, smooth, and normally distributed latent representations in which close latent points correspond to very similar reconstructions. Previous studies in the literature demonstrate the superior performance of the VAE over the conventional AE in AD [103, 307, 352].

The AD model training process comprises data preprocessing and model training (see Fig. 4.3). We prepared the modeling data sets after master-slave configuration adjustments, data cleaning, and data standardization. The control cards—J14 and J15—of the ngCCM operate in a master-slave configuration; consequently, some sensors depend on the configuration. The input preprocessing adjusts the variables as "MASTER" and "SLAVE" after identifying the master card from the signal variations. Obtaining clean data for the healthy instances and reducing contamination in the training data is one of the main challenges of semi-supervised

[1] https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html

[2] https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

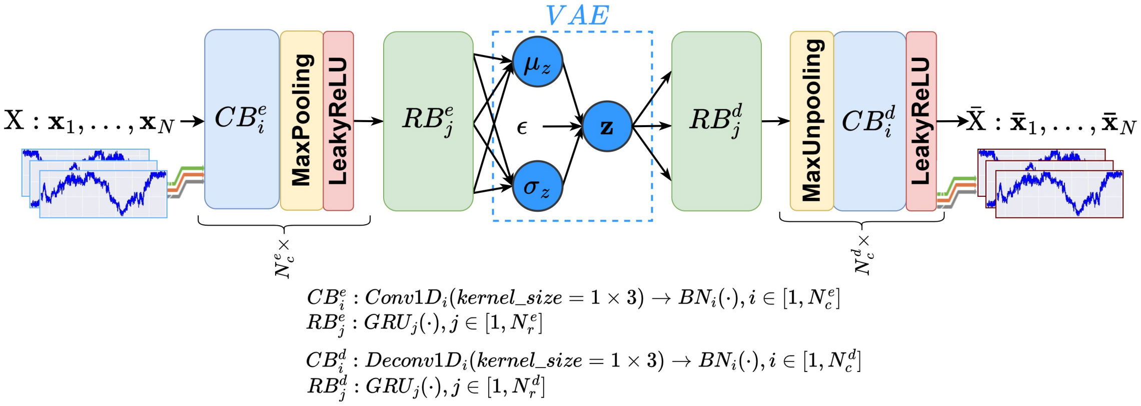

VAE
$\mathbf{X} : \mathbf{x}_1, \ldots, \mathbf{x}_N$
CB_i^e
MaxPooling
LeakyReLU
RB_j^e
μ_z
ϵ
σ_z
$\mathbf{z}$
RB_j^d
MaxUnpooling
CB_i^d
LeakyReLU
$\bar{\mathbf{X}} : \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N$
$N_c^e \times$
$N_c^d \times$
$CB_i^e : Conv1D_i(kernel\_size = 1 \times 3) \rightarrow BN_i(\cdot), i \in [1, N_c^e]$
$RB_j^e : GRU_j(\cdot), j \in [1, N_r^e]$
$CB_i^d : Deconv1D_i(kernel\_size = 1 \times 3) \rightarrow BN_i(\cdot), i \in [1, N_c^d]$
$RB_j^d : GRU_j(\cdot), j \in [1, N_r^d]$

Figure 4.2: Autoencoder architecture: the CNN blocks (CB^e and CB^d) employ pipeline of three networks $N_c^d = N_c^d = 3$—each consists 1DCNN with 32 kernels, BN for network weight regularization, LeakyReLU for non-linear activation and MaxPooling1D and MaxUnpooling1D for data translation insensitive feature retrieval and reconstruction unsampling in the encoder and decoder, respectively. The RNN blocks consist of two layers $N_r^e = N_r^d = 2$—RB^e : GRU(8) → GRU(2) and RB^d : GRU(2) → GRU(8). The VAE $\mathbf{z} = \mu_z + \sigma_z \odot \epsilon$ uses fully-connected linear networks for μ_z and σ_z. The $\epsilon \sim \mathcal{N}(0, I)$ and $\odot$ signify an element-wise product.

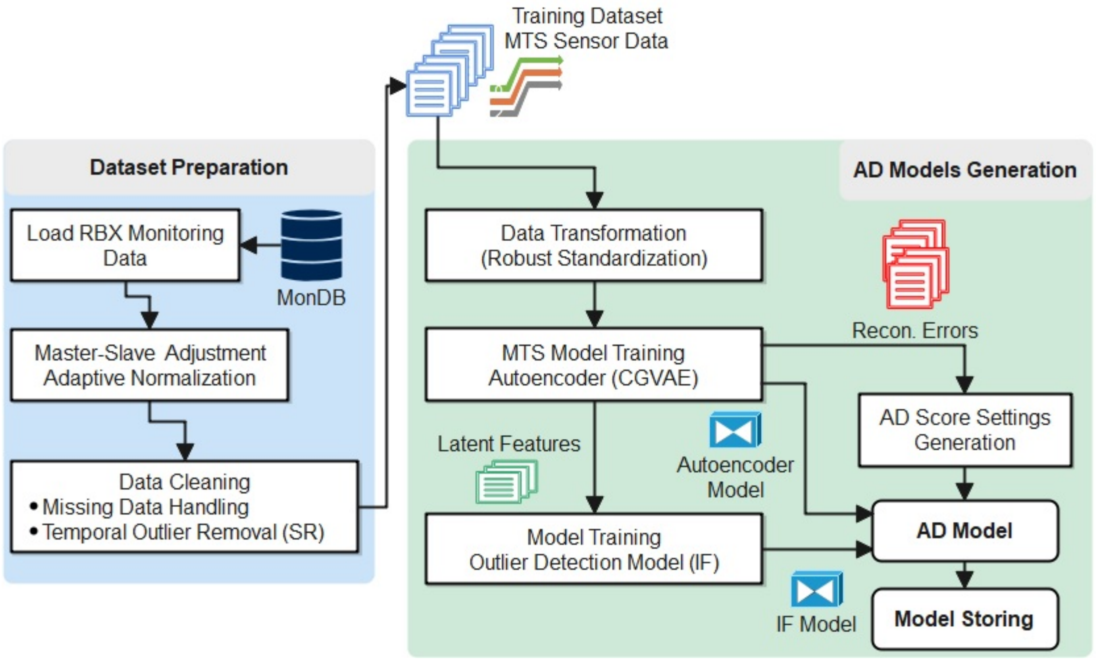


Figure 4.3: Model training workflow of the proposed AD system. MonDB: database for monitoring sensor data, MTS: multivariate time series, SR: saliency mapping spectral residual for univariate time series data, and IF: isolation forest.

learning [99]. We utilize a spectral residual algorithm (SR)—a state-of-the-art and lightweight univariate TS outlier detection method—to clean potential outliers from each univariate sensor reading in the training set [95,148]. We standardize the multivariate data (into $\mu = 0$, $\sigma = 1$) using [25%, 75%] quantiles to avoid sensitivity to any leaked outliers. The sensor measurements across different RBXes have some discrepancies—e.g., DC-shifting due to offset voltage variations or maintenance. We employ an adaptive normalizer that detects transitions between stable states using an online edge-detection algorithm based on Canny detector [353]—applies convolution between a derivative of Gaussian kernel with the input signal to detect robust signal edges and smoothing noise (see Algorithm 1). The adaptive normalizer estimates the localized median values for each time segment between detected edges. This mitigates false-positive transitions—due to changes in the operating center after maintenance or reconfiguration—the heterogeneity among sensor signals and local variations among RBXes. The normalization enables us to incorporate data from different RBX systems into the training dataset and train a single model for all RBXes; it also relieves the need for large training data sets that comprehensively cover all possible scenarios, which is impractical in most cases.

Algorithm 1 Time series signal adaptive normalizer

```
procedure TimeSeriesAdaptiveNormalizer($\mathbf{x}, \tau, k_d$)
▷ $\mathbf{x}$ is the input univariate TS signal          ▷ $\tau$ is the minimum signal segment length
    $\mathbf{x}_f \leftarrow SignalSmoothing(\mathbf{x})$          ▷ smooth and clean signal using sliding median filter
    $\mathbf{x}_e \leftarrow SignalEdgeDetection(\mathbf{x}_f)$          ▷ generate signal transition edges
    $\mathbf{s} \leftarrow [\,]$          ▷ placeholder for normalized signal
    for $\mathbf{y}, l_y \in$ $GetTimeSegemnt(\mathbf{x}_e)$ : do          ▷ TS segments between consecutive edges
        if $l_y > \tau$ then
            $\mathbf{y}_n \leftarrow GetInitialStableReading(\mathbf{y}, \tau)$          ▷ get stable signal segment for normalization
        else
            $\mathbf{y}_n \leftarrow \mathbf{y}$
        end if
        $\bar{\mathbf{y}} \leftarrow \mathbf{y}/MEDIAN(\mathbf{y}_n)$          ▷ normalized signal segment
        $\mathbf{s} \leftarrow Append(\mathbf{s}, \bar{\mathbf{y}})$          ▷ stable signal for normalization
    end for
return s
end procedure
```

We utilize mean absolute error (MAE) as the reconstruction loss function $\mathcal{L}_{rec}$ regulized with KL divergence loss to achieve variational latent space to the AE:

$$\begin{aligned} \mathcal{L}_{rec} &= \mathbb{E}\left[|\mathrm{X} - \bar{\mathrm{X}}|\right] \\ \mathcal{L}_{KL} &= -D_{KL}\left[\mathcal{N}(\mu_z, \sigma_z), \mathcal{N}(0, I)\right] \\ \mathcal{L} &= \mathcal{L}_{rec} + \beta\mathcal{L}_{KL} + \rho\|W\|_2^2 \end{aligned} \tag{4.8}$$

where the $\mathbb{E}$ is a mean operator. The $\mathcal{L}_{rec}$ is the MAE reconstruction loss, $\mathcal{L}_{KL}$ is the KL divergence loss with tunable regularization parameter β, and the last term of the $\mathcal{L}$ is L_2 weight regularization with a parameter of ρ. We trained the AE using *Adam* optimizer using a learning rate of 10^{-4}, $\beta = 10^{-3}$, and $\rho = 10^{-7}$ on 5000 epochs with early-stopping; we trained the IF outlier detection model on the extracted latent vectors of the training dataset after training the AE.

Anomaly Detection Explanation: The AE generates reconstructed signals and low-dimensional latent features from input signals with a time length of T. The AD system calculates reconstruction and latent anomaly scores to detect anomalies. Feature attribution algorithms render the anomaly explanations. We present a multidimensional anomaly output explanation based on a combination of feature attributions across different stages (see Fig. 4.4).

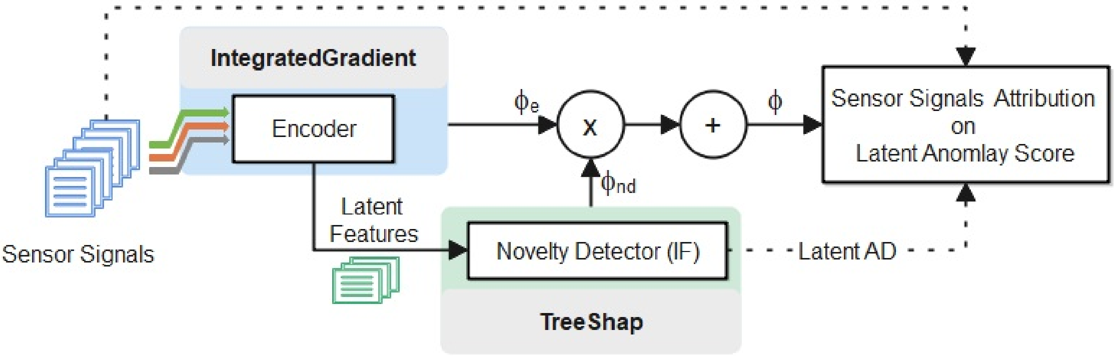


Figure 4.4: System AD anomaly score explanation via feature attribution estimation using *Integrated Gradient* and *TreeShap*. FA from TS input variables to latent AD score.

Explanation of AD decision after detecting anomaly is essential to facilitate the applicability of data-driven AD models [336]. There is limited effort in the literature on post-hoc model explanation for TS models [297–300]. Refs. [297, 300] employ CAM approach—that require a global pooling layer—for widely used in classification CNN models. Ref. [298] utilizes SHAP [221] to explain the output of a gradient-boosted ML model applied to time series medical data. The modeling approach was not based on the raw TS data; the TS data was transformed into non-temporal statistical features using time windowing before modeling. Similarly, Ref. [336] engages SHAP for AD explanation on a non-temporal autoencoder model. Gradient methods such as IG [85] provide good accuracy and are computationally faster than KernelSHAP and DeepSHAP for differentiable DL models [299, 341]; IG is not applicable for models with non-differentiable prediction surfaces—e.g., tree-based models [342].

We present an explanation aggregation mechanism to adopt post-hoc feature attribution (FA) methods for the TS AD model. We integrate multiple explanation methods to fully explain detected latent anomalies s_{LD} since the CGVAE AD model consists of a deep learning AE model and a tree-based IF algorithm. We employ IG [85] to estimate the feature attribution $\phi_e(\mathrm{X}, \mathbf{z})$ of the sensor inputs $\mathrm{X} \in \mathbb{R}^{N_x \times T}$ on the encoded latent features $\mathbf{z} \in \mathbb{R}^{N_z \times 1}$; this generates FA $\phi_e(\mathrm{X}, \mathbf{z}) \in \mathbb{R}^{N_z \times N_x \times T}$, where N_z, N_x, and T are the size of latent feature dimension, number of the input sensors and size of the time window, respectively. The TreeSHAP [84] estimates the FA $\phi_{nd}(\mathbf{z}, s_{LD}) \in \mathbb{R}^{N_z \times 1}$—the attribution of the latent features $\mathbf{z}$ on the AD anomaly score output s_{LD} of the IF. The end-to-end temporal FA $\phi(\mathrm{X}, s_{LD}) \in \mathbb{R}^{1 \times N_x \times T}$ is the aggregated ϕ_e after weighted by the ϕ_{nd} as:

$$\bar{\phi}(X, s_{LD}) = \sum_{l=1}^{N_z} \phi_e(\mathrm{X}, z_l) \times \phi_{nd}(z_l, s_{LD}) \tag{4.9}$$

We finally rank the input sensors based on a final aggregate FA scores $\phi(\mathrm{X}, s_{LD}) \in \mathbb{R}^{N_x \times 1}$ over the entire time window—exploiting the additive property of the FA scores as:

$$\phi(\mathrm{X}, s_{LD}) = \sum_{t=1}^{T} \bar{\phi}(x_t, s_{LD}) \tag{4.10}$$

4.1.3 Experimental Results and Discussion

We trained the AD model on one-month data—01 – 31/10/2018–from three stable RBXes—HEM01, HEM07, and HEP11—with relatively low outlier contamination—and evaluated the performance for the entire date range of three months—15/09 – 10/12/2018—on all the 34 RBXes. We set $T = 4$ hours sliding time window with 30-minute steps for the anomaly score estimation. We heuristically determined the $k_i = 15\ \forall i$, $K_{LD} = 0.8$, $\alpha_s = K_{MD} = 15$ and $K_{AFS} = 0$ to estimate the detection decision thresholds.

The distributions of the extracted low-dimensional representation latent features of the RBXes are illustrated in Fig. 4.5. The AE maps the distribution of the regular instances into the center as forced by the variational layer of the VAE. Fig. 4.6 portrays the distributions of the employed anomaly scores relative to the latent features for HEM01. The plot illustrates the clustered correlation between the latent and the anomaly scores—stronger anomalies are farther from the center. Some anomalies still overlap with normal operations near the center but with high reconstruction errors—demonstrating the need to employ latent and reconstruction-based metrics to leverage an AD system (see Section 4.1.2).

The HCAL experts utilize error flag counter quantities to monitor the RBX health status. We thus generated anomaly labels from the most relevant error counters of the ngCCM (see Table 4.1) to evaluate the capability of the AD model. The error counters relate to faults in communication, backend power, and digitization electronics. The anomaly labels were produced for incremental changes on any of the counters. We utilize the widely used classification metrics such as *precision* (P), *recall* (R), and *f1-score* (F_1) metrics for the evaluation:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, F_1 = \frac{2 \times P \times R}{P + R} \tag{4.11}$$

where TP is true positive, FP is false positive, and FN is false negative; the true class represents the anomaly.

Fig. 4.7 portrays the classification performance on detection accuracy of the AD model. Fig. 4.8 depicts the contribution of each sensor from the sensor AD. The results demonstrate the robustness of the AD model in capturing most of the potential anomalies that are flagged by the error counters across the RBXes (see

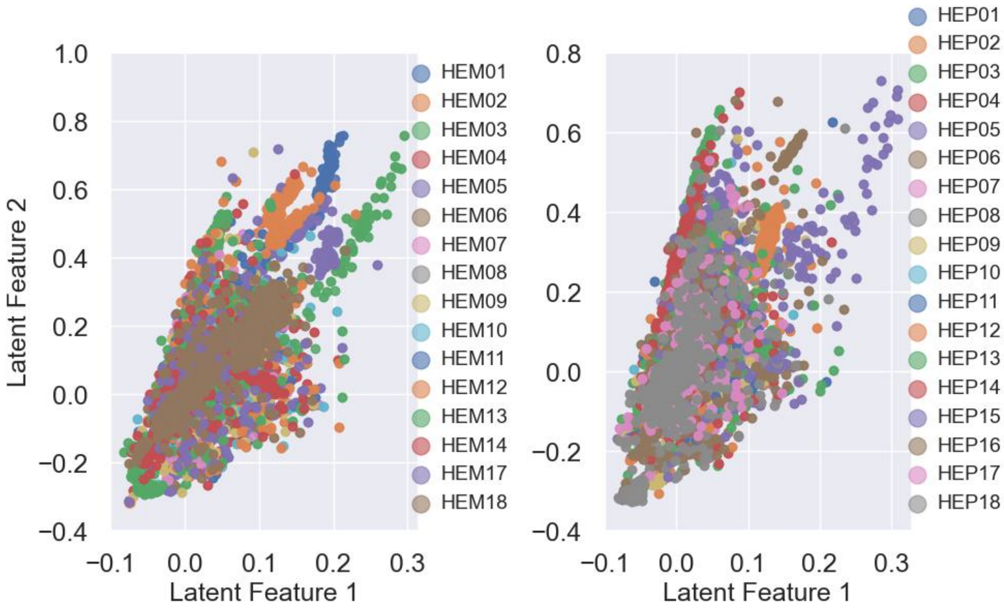


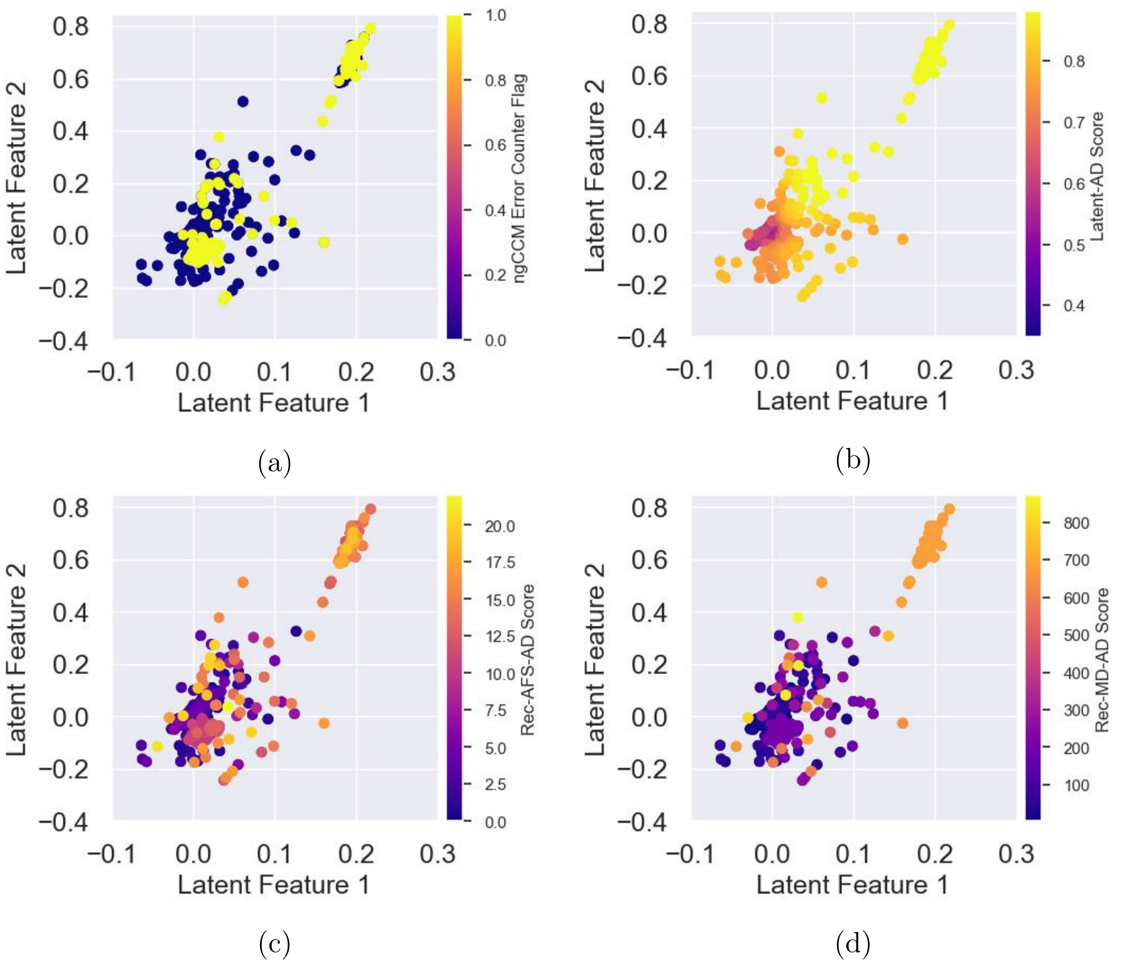
Figure 4.5: Distribution of the latent features of all RBXes: (left) for the HEM, and (right) for the HEP.

Table 4.1: Error counter variables of the ngCCM of the HE subdetector.

Counter Category	**Error Counter Variables**
B2B	B2B_RX_PLL_LOCK_LOST_CNT, B2B_RX_PRBS_ERROR_CNT, B2B_RX_RSDEC_ERROR_CNT
FEC	FEC_BKP_PWR_FLIP_CNT, FEC_DV_DOWN_CNT, FEC_RX_PRBS_ERROR_CNT
FLAG	FLAG_FILTERED-CG, FLAG_FILTERED-LG, FLAG_FILTERED-PG
MEZZ	MEZZ_QIE_RESET_MISSING, MEZZ_RX_PLL_LOCK_LOST_CNT, MEZZ_RX_PRBS_ERROR_CNT, MEZZ_RX_RSDEC_ERROR_CNT, MEZZ_TMR_ERROR_COUNT

Fig. 4.7). The low precision and consequently diminished F_1 scores in some RBXes suggest that the AD model produced more alerts that can be either false flags or potential anomalies missed by the error counters.

We have found most of the FPs are actual outliers and potential anomalies. The error counters (given in Table 4.1) have partial correlations only with some sensor quantities, which monitor the backbone power and communication, and anomalies from the remaining sensor are not covered. Not all the anomaly flags from the error counters are thus manifested in the sensor data and vice-versa. Most detected anomalies in FPs correspond to a change in temporal fluctuation patterns, gradual decay, and ramping-up in some sensors. Since we have excluded the error counters for the slave ngCCM control card during label generation due to stability issues on their values, isolated anomalies from the slave card sensors are classified as false flags in HEM05, HEM13, HEP05, HEP13, and HEP14. Fig. 4.9 demonstrates the F_1 noticeably improves for most RBXes with detection threshold α_s from 5 to 35—sensitivity reduces—demonstrating the high-level ngCCM performance error



(a)

(b)

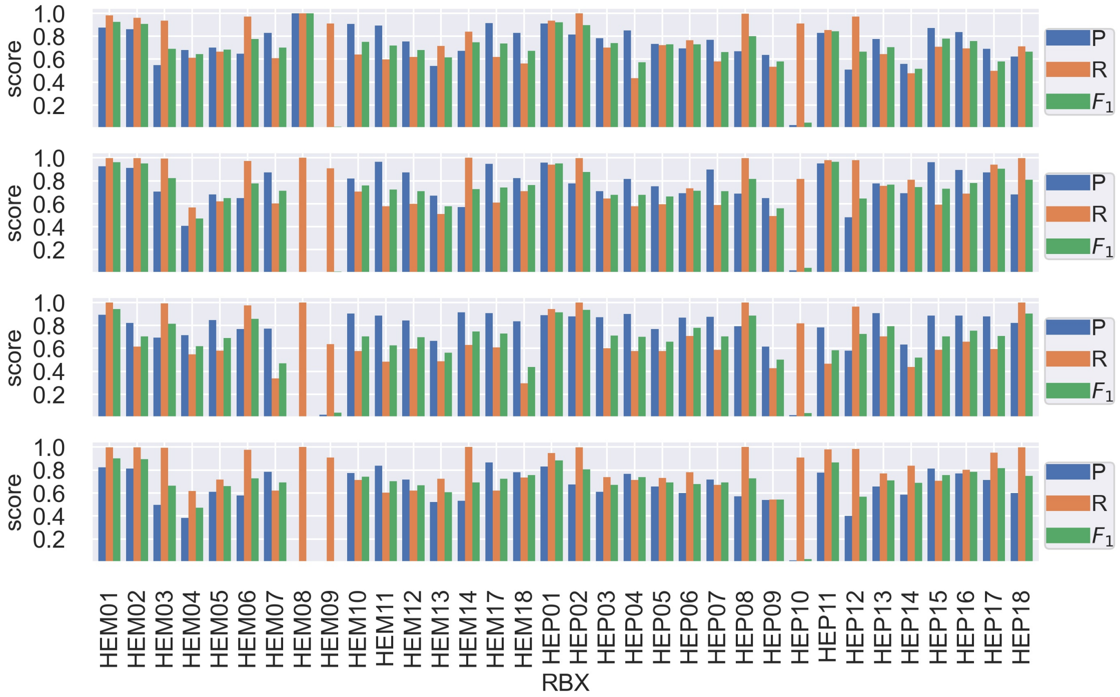
(c)

(d)

Figure 4.6: The representational latent features and anomaly scores of RBX-HEM01: a) generated anomaly flags from error counters of the RBX, b) latent-AD IF novelty score, c) Rec-AFS-AD score, and d) Rec-MD-AD score. From b to d portray anomaly scores, and the color bars show the strength.

counters correlate with the stronger sensor anomalies of the low-level electronic components; the F_1 score increases from 0.76 into 0.78 that the precision of detection increases—lowering FP—with limited impact on the recall—TP—as the α_s increases up to 30.

We present an ablation study to demonstrate the relevance of the model-building network blocks of the proposed CGVAE model. The convolutional and variational networks are removed one by one from the CGVAE model: 1) CGVAE- the proposed AD model, 2) CGAE- CNN-GRU AE without the variational layer, and 3) GAE- a GRU AE without the convolutional and variational layers; it uses $\mathrm{X}(28) \rightarrow \mathrm{GRU}(32) \rightarrow \mathrm{GRU}(32) \rightarrow \mathrm{GRU}(2) \rightarrow \mathrm{GRU}(32) \rightarrow \mathrm{GRU}(32) \rightarrow \mathrm{GRU}(28) \rightarrow \bar{\mathrm{X}}(28)$ to avoid under-fitting. Table 4.2 summarizes performance on the F_1 score. The ablation study shows that the modeling blocks contribute to the performance gains of the proposed CGVAE AD model. The CNNs stacked with the GRUs significantly enhanced the learning performance compared to the GAE. The variational low-dimensional latent from the CGVAE has achieved a relatively higher score on



(top to bottom) Latent-AD- (A_{LD}), Rec-AFS-AD (A_{ASF}), Rec-MD-AD (A_{MD}), and Latent-Rec-AD (A_{MTS})—union of all and improved the recalls or correct detection rate.

Figure 4.7: AD classification performance on different anomaly scoring approaches. Low precision on the HEM08, HEM09, and HEP10; HEM08 has a sensor with missing reading, and HEM09 and HEP10 have counter quantity log issues—failed to record errors—and delivered fewer flags for comparison.

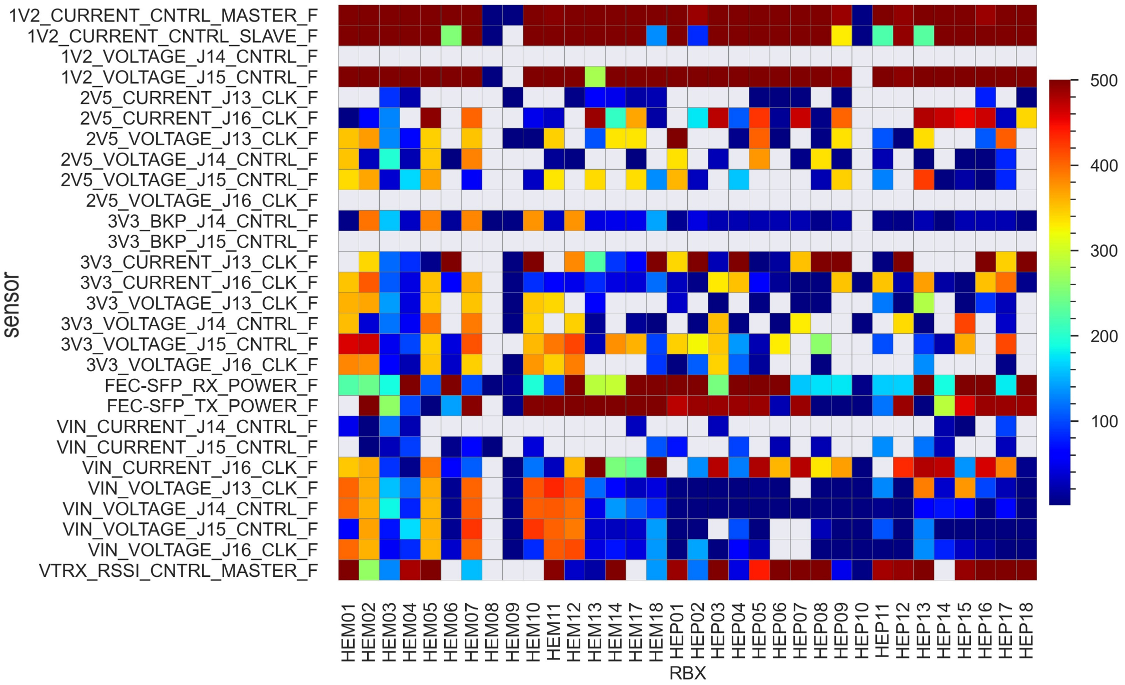


Figure 4.8: Sensor AD reconstruction TP anomaly flags. The color bar shows the flag counts threshold at 500.

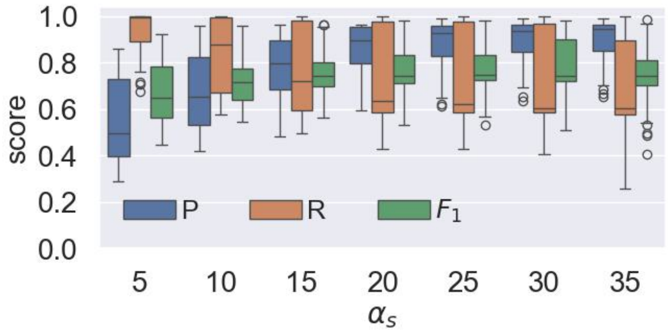


Figure 4.9: Rec-AFS-AD performance with varied detection threshold α_s values.

the latent AD—improving the overall AD.

Table 4.2: Ablation study on proposed AD model—F_1 score.

Model	Latent-AD	Rec-ASF-AD	Rec-MD-AD	Latent-Rec-AD
CGVAE	**0.725**	**0.763**	0.716	**0.727**
CGAE	0.633	0.717	0.747	0.683
GAE	0.592	0.714	**0.758**	0.616

CGAE: without the variational layers.

GAE: without the convolutional and variational layers.

The scores are averaged over all the RBXes except HEM04, HEM08, HEM09, and HEP10 since the label reference—error counter variables—did not adequately capture the actual outliers.

We chose RBX-HEM04 operation from 15 – 30/11/2018 to demonstrate the feature attribution explanation of the AD system. The time window includes various types of anomalies, such as spikes, dips, long trends, and multivariate anomalies (see Fig. 4.10). Fig. 4.11 illustrates correlations among the latent features, anomaly scores, and flags generated from the ngCCM error counters.

The model explainer proposes the topmost influential input sensors with attribution strength on the temporal data points for a given system anomaly from the latent AD. Fig. 4.12 presents the results returned by the model explainer for a given detected anomaly. Fig. 4.13 provides more anomaly samples with generated ranked explanations compared to the sensor reconstruction AD results. The high agreement—a high hit rate—between per-sensor AD and the generated explanation results demonstrate the promising effectiveness of our proposed AD explainer approach for TS AD models.

4.1.4 Summary

We have presented our unsupervised anomaly detection model for the next-generation clock and control system of the HCAL Endcap calorimeter. Our AD model aims to achieve efficient localized feature extraction, temporal learning, and descriptive abnormality representation of multidimensional inputs. The study has demonstrated the effectiveness of using latent and reconstruction anomaly scores in improving AE-based AD. We have successfully combined feature attribution scores to provide output explanations of the AD system incorporating different machine

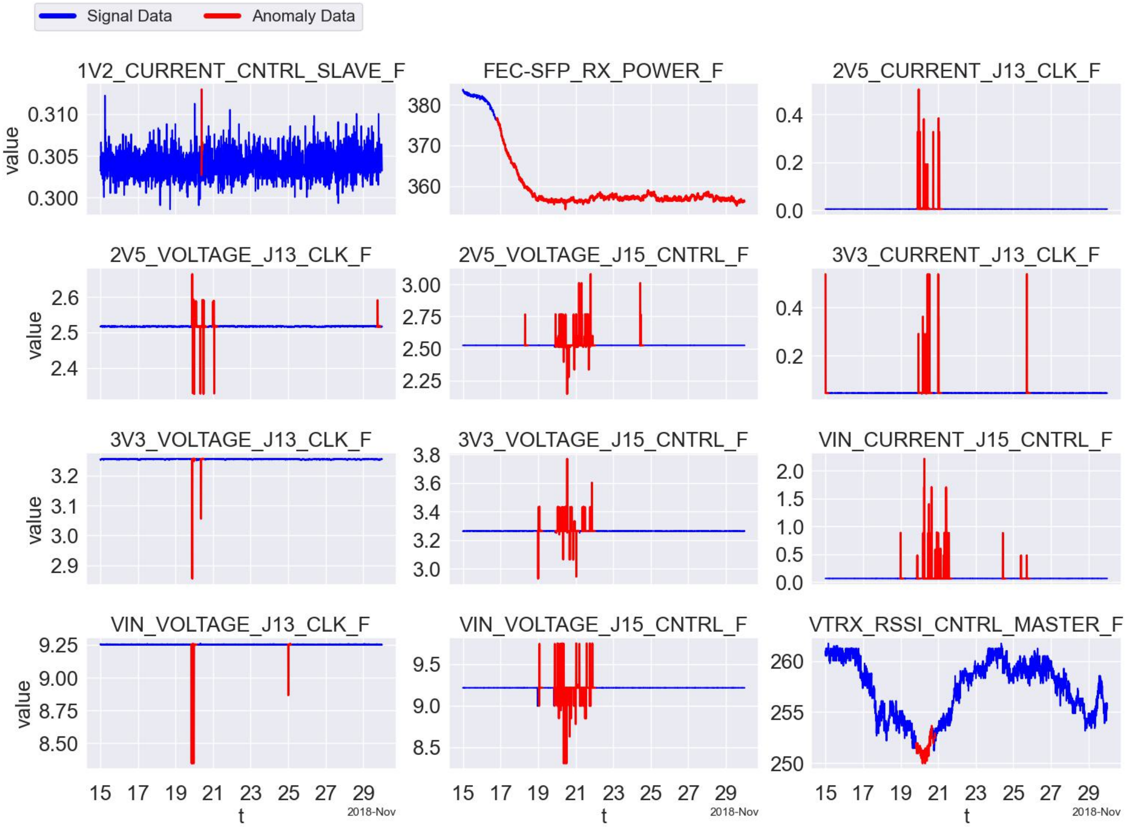


Figure 4.10: Multivariate sensors of HEM04 with marked anomalies. Only sensors with anomaly reports are depicted. Most sensors have global temporal outliers and the FEC-SFP_RX_POWER_F sensor exhibits a trend drifting anomaly starting on 17/11/2018.

learning models. Our study has revealed that the further relevant health status of the ngCCM can be inferred from the automated monitoring of diagnostic sensor signals—leveraging the existing ngCCM error counter variables. The developed AD system will thus assist the HCAL in actively monitoring and debugging various types of anomalies of the RBXes.

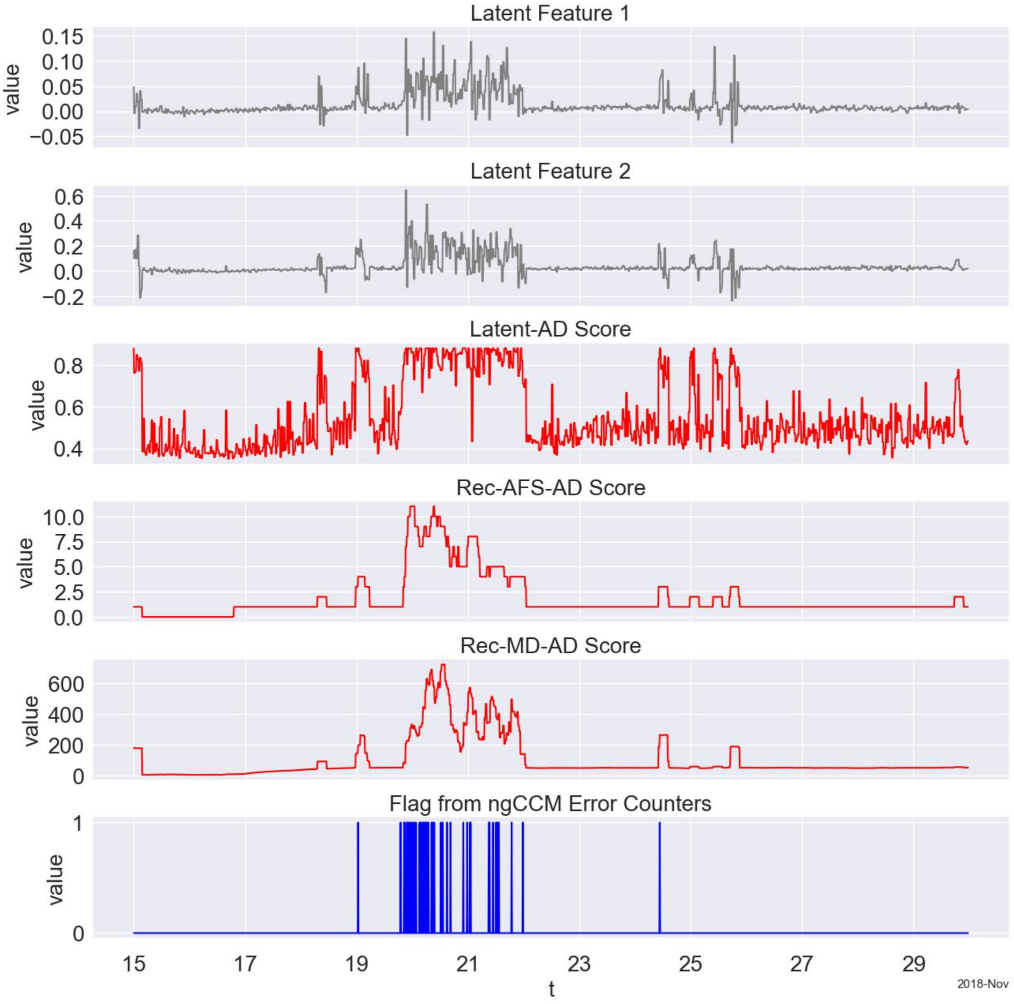
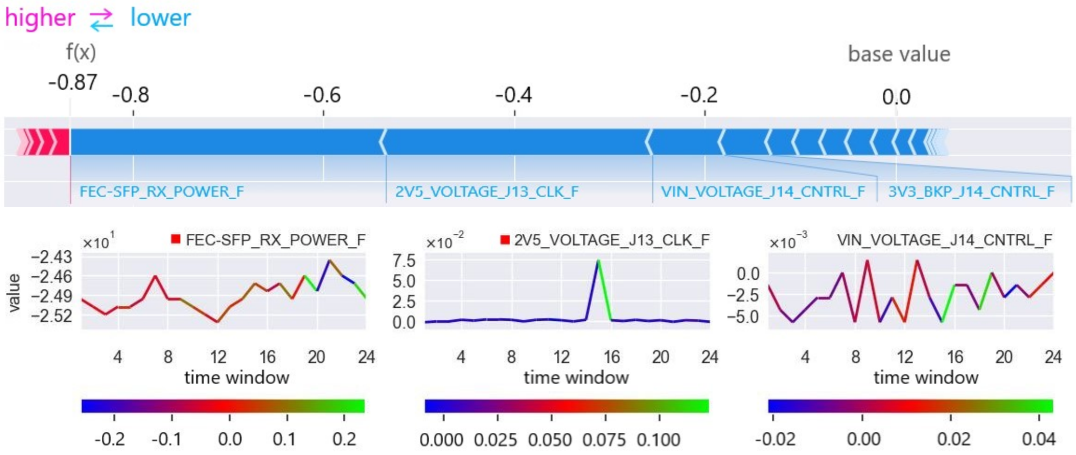
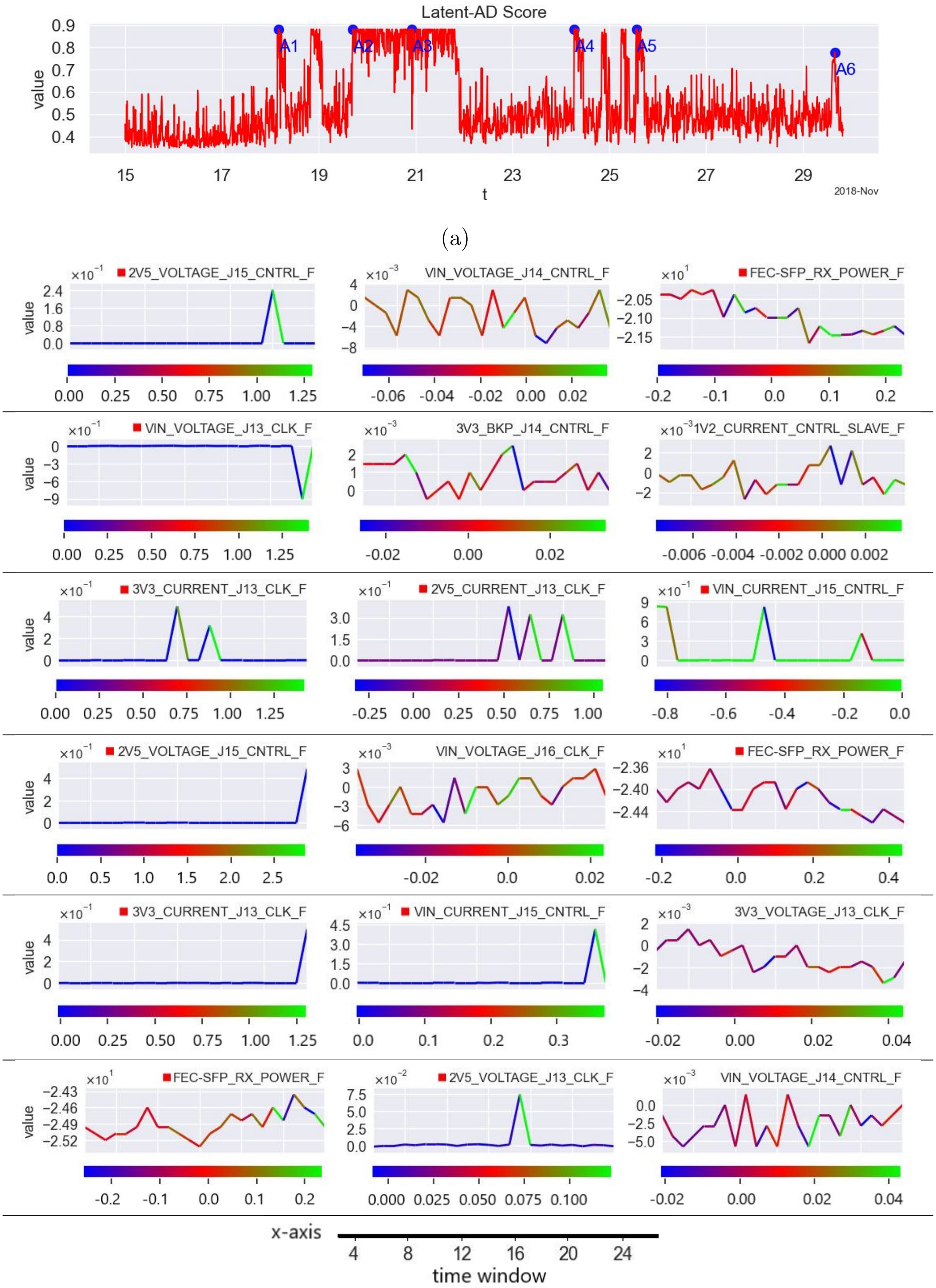


Figure 4.11: Latent features and system anomaly scores on RBX-HEM04. Gradual system drift during 17–30/11/2018 of the FEC-SFP_RX_POWER_F was not detected by the error counters of the ngCCM (bottom plot), but our AD model has captured it (see 3^{rd} – 5^{th} plots).



The y-axis value is a deviation from the nominal value of the sensors.

Figure 4.12: AD explainer on a detected latent anomaly at 29/11/2018 15:40: (top) illustrates the ranked influential input signals with the bar sizes designating the contributions, and (bottom) depicts signal sections of the top three attributing sensors—ranked from left to right—and the color bar shows the attribution strength with a sign denoting the direction. The top-two sensors with strong attributions have also reported univariate reconstruction anomalies, indicated by the red marker on the title (see Fig. 4.10 and *A6* in Fig. 4.13).



(a)

(b)

Figure 4.13: Explanation on latent AD anomalies of RBX-HEM04: a) sample anomalies—*A1* at 18/11 04:10, *A2* at 19/11 16:50, A3 at 20/11 22, *A4* at 24/11 06:30, *A5* at 25/11 13:20, and *A6* at 29/11 15:40—and b) influential sensors from our AD explainer, and the red-marker on the plots' title indicates reconstruction sensor AD flag at the same timestamp.

4.2 Spatio-Temporal Anomaly Detection with Graph Networks for Data Quality Monitoring of the Hadron Calorimeter

The CMS experiment employs an online data quality monitoring (DQM) system to promptly spot and diagnose particle data acquisition problems to avoid data quality loss. Several studies have proposed ML models to leverage DQM automation; only a few have dealt with temporal models to underpin system monitoring. ML monitoring of DQM of the HCAL—relatively unexplored—poses a multidimensional challenge due to the calorimeter's depth-wise segmentation. We present semi-supervised spatio-temporal AD monitoring (GraphSTAD) for the physics particle sensing channels of the HCAL using three-dimensional particle hit map data of the DQM. The GraphSTAD employs CNNs and GNNs to learn local spatial characteristics induced by particles traversing the detector, and global behavior because of shared backend circuit connections and housing boxes of the channels, respectively. The RNNs capture the temporal evolution of the extracted spatial features. We validate the accuracy of the proposed AD system in capturing abnormal channels in the LHC collision data sets. The system performs production-level accuracy and is integrated into the CMS DQM production system—for real-time monitoring of several anomaly types. We provide a performance comparison with alternative benchmark models to demonstrate the promising leverage of the presented system.

The DQM system aims to guarantee high-quality physics data through online monitoring that provides live feedback during data acquisition and offline monitoring that certifies the data quality after offline processing [49]. The *online DQM* identifies emerging problems using reference distribution and predefined tests to identify known failure modes [71, 72] using summary histograms, such as a digi-occupancy map of the CMS calorimeters. A digi-occupancy map contains the histogram record of particle hits of the data-taking channels of the calorimeters. The CMS calorimeters may encounter problems during data taking—including issues with the frontend particle sensing scintillators, backend hardware, and algorithms, that might appear in the digi-occupancy maps. The growing complexity of detectors and the variety of physics experimentation make data-driven AD systems essential tools for CMS to identify and localize detector anomalies automatically. Recent efforts in CMS have proposed DL for AD applications for the DQM [39, 40, 49, 50]. The synergy in AD has thus far achieved promising results on spatial 2D histogram maps of the DQM for the ECAL [39] and the muon detectors [40]. Previous studies only considered extreme anomalies—no reading dead and high noise hot calorimeter channels. Detecting degrading channels is essential for quality deterioration monitoring and early intervention, but they are often challenging to capture; for instance, the improperly tuned bias voltage on the HCAL physics particle sensing channels caused non-uniformity in the hit map of the DQM, but the channels were neither dead nor hot [75]. The calorimeter channels may degrade with subtle abnormality before reaching extreme channel fault status. Capturing such subtle anomalies—e.g., slow system degradation—makes temporal AD models appealing for early anomaly prediction before ultimate system failure. Time-aware models extract temporal context to enhance AD performance. A few efforts have thus far been focused on temporal models despite the acknowledged potential in the future automation technology challenges at the LHC [39,52]. Our study focuses on DQM automation through time-aware AD modeling using 3D histogram maps of the HCAL. The digi-occupancy data of the HCAL is 3D due to its depth-wise calorimeter segmentation. It poses multidimensional challenges and it is relatively unexplored in ML endeavors. The particle hit map data of the HCAL are highly dependent on the collision luminosity—a measure of how many collisions are happening in a particle accelerator—and the number of particles traversing the calorimeter. The effort on normalization that enhances learning generalization of ML models is still limited.

We address the above gaps while investigating the performance enhancement of temporal AD DL models for the HCAL DQM. We propose to detect anomalies of the HCAL particle sensing channels through a semi-supervised ST AD system—GraphSTAD—from spatial digi-occupancy maps of the DQM. Anomalies can be unpredictable and come in different patterns of severity, shape, and size—often limiting the availability of labeled anomaly data covering all possible faults. We employ a semi-supervised approach for the AD system; the concept for the AD is that an autoencoder (AE)—trained to reconstruct healthy digi-occupancy maps—would adequately reconstruct the healthy maps, whereas it yields high reconstruction error for maps with anomalies. Since abnormal events can have spatial appearance and temporal context, it is desirable to combine both the spatial and temporal features for ST AD [109,111,113,116–121,142]. Our GraphSTAD system is a graph-leveraged spatio-temporal AD system that detects various channel anomalies. The spatial nature of the digi-occupancy map of the HCAL may exhibit irregularity; adjacent channels with Euclidean distance are exposed to collision article hits around their region, but the channels may belong to different backend circuits—resulting in non-Euclidean spatial behavior on the digi measurements. The GraphSTAD system captures the behavior of channels from regional collision particle hits and electrical and environmental characteristics due to a shared backend circuit of the channels to detect the degradation of faulty channels effectively. The AD system attains these utilities using a deep AE model that learns spatial behavior, physical connectivity-induced shared behavior, and temporal behavior using CNN, GNN, and RNN, respectively.

We evaluate our proposed AD approach in detecting spatial faults and temporal discords on digi-occupancy maps of the HCAL. We simulate different types of synthetic anomalies—*dead channels* without registered hits, *hot channels* dominated by electronic noise—resulting in a much higher hit count than expected, and *degraded channels* with deteriorated particle detection efficiency—resulting in lower hit counts than expected—to analyze the effectiveness of the AD model. The results demonstrate promising performance in detecting and localizing the anomalies. We further validate the efficacy in detecting real anomalies and discussed comparisons to benchmark models and the existing DQM system.

4.2.1 Dataset Description

We employed digi-occupancy data of the online DQM system to train and validate the proposed AD system. The digi-occupancy data is 3D spatial maps with $i\eta$, $i\phi$, and *depth* axes, and contains digi histogram records of the physics readout channel sensor of the calorimeter referenced by $i\eta = [-32, \ldots, 32]$, $i\phi = [1, \ldots, 72]$ and $depth = [1, \ldots, 7]$ axes (see Fig. 4.14). A digi—also called hit—is a reconstructed and calibrated signal response of a particular *ieta*, *iphi*, and *depth* segment of the calorimeter.

Several errors can arise in the calorimeter affecting the frontend particle sensing scintillators, the digitization and communication systems, the backend hardware, or the algorithms. These errors appear in the digi-occupancy map as holes, under- or over-populated, or saturated bins. The value of the digi-occupancy varies with the received luminosity—the recorded by CMS and hereafter referred to as the luminosity—and the number of events—particles traversing the calorimeter. The maps from a sequence of lumisections (LSs) constitute attribution of ST data with correlated spatial and temporal relations [141].

The digi-occupancy root file data sets were collected in 2018 during the LHC Run-2 collision by the CMS experiment. The data set—from the *ZeroBias primary dataset*—contains approximately 20K LSs from 20 healthy runs pre-scrutinized by the CMS certifiers, and declared in the "Golden JSON" of the DQM as of good quality [354]. The maps—one per LS—were populated with the per LS received luminosity up to 0.4 pb^{-1}, and the number of events up to 2250. The digi-occupancy maps from a sequence of LSs constitute attribution of ST data with correlated spatial and temporal relations [141]. Our working ST data set contains 20K map samples—each with a dimension of $[i\eta = 64 \times i\phi = 72 \times depth = 7]$).

4.2.2 Methodology

There is a lack of adequate labeled anomaly data covering all possible channel fault scenarios for the HCAL, and the anomalies may follow unpredictable patterns with different severity, shape, and size. We thus employ a semi-supervised approach for the AD system—GraphSTAD system; we trained a deep AE model to reconstruct healthy digi-occupancy maps with low contamination of anomalies. We present an ST reconstruction AE to detect abnormality in the HCAL channels using reconstruction deviation scores on ST digi-occupancy maps from consecutive lumisections (see Fig. 4.15). The AE combines CNNs, GNNs, and RNNs to capture ST characteristics of digi-occupancy maps. The spatial feature extraction of the CNNs is leveraged with GNNs to learn circuit and housing connectivity-induced spatial behavior irregularities among the HCAL sensor channels. There are approximately 7K channels—pixels—on the digi-occupancy map of the HCAL Endcap subsystem—housed in 36 RBXes. The channels in a given RBX are susceptible to system faults in the RBX due to the shared backbone circuit and environmental factors like temperature and humidity. Our proposed GraphSTAD employs GNNs—in its spatial feature extraction network pipeline—to capture the characteristics of the HCAL

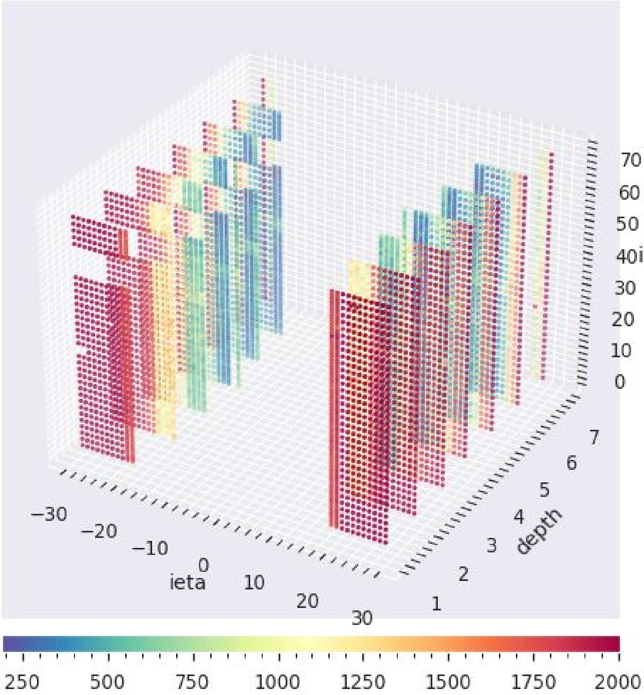


Figure 4.14: Digi-occupancy map (*year=2018, RunId=325170, LS=15)* of the HE. The HE channels are placed in $|i\eta| = [16, 29]$, $i\phi = [1, 72]$, and $depth = [1, 7]$ spatial coordinates of the DQM. Each pixel in the map corresponds to the readout channel of the calorimeter after a particle hit. The HCAL covers a considerable volume of the CMS detector and has segmentations along three axes ($i\eta$, $i\phi$, and $depth$). The missing section near the top-left of the map is due to two failed RBX (HEM15 and HEM16) sectors during the 2018 collision runs.

channels owing to their shared physical connectivity to a given RBX. GNNs have recently achieved promising results in several applications at the LHC [23, 284] and outperformed CNNs in learning irregular calorimeter geometry [285] and in pileup mitigation [38]. The GraphSTAD system exploits both CNNs and GNNs [286,287] to capture Euclidean and non-Euclidean spatial characteristics of the HCAL channels, respectively.

The data preprocessing stages of our proposed approach involve two main tasks: 1) digi-occupancy map renormalization, and 2) graph-adjacency matrix generation.

Digi-occupancy Map Renormalization: The digi-occupancy (γ) map data of the HCAL varies with the received luminosity (β) and the number of events (ξ) (see Fig. 4.16). We devise a renormalization of the γ through a regression model $\mathcal{R}$ to have a consistent quantity interpretation of the γ and build an AD model that robustly generalizes previously unseen run settings—β and ξ variations. The $\mathcal{R}$ estimates the renormalizing $\bar{\gamma}_s$ at the s^{th} LS using β and ξ as:

$$\bar{\gamma}_s = \mathcal{R}(\xi, \beta) \tag{4.12}$$

The model $\mathcal{R}$ is trained to minimize the MSE cost function, $\mathbb{E}[(\gamma_s - \bar{\gamma}_s)^2]$, where

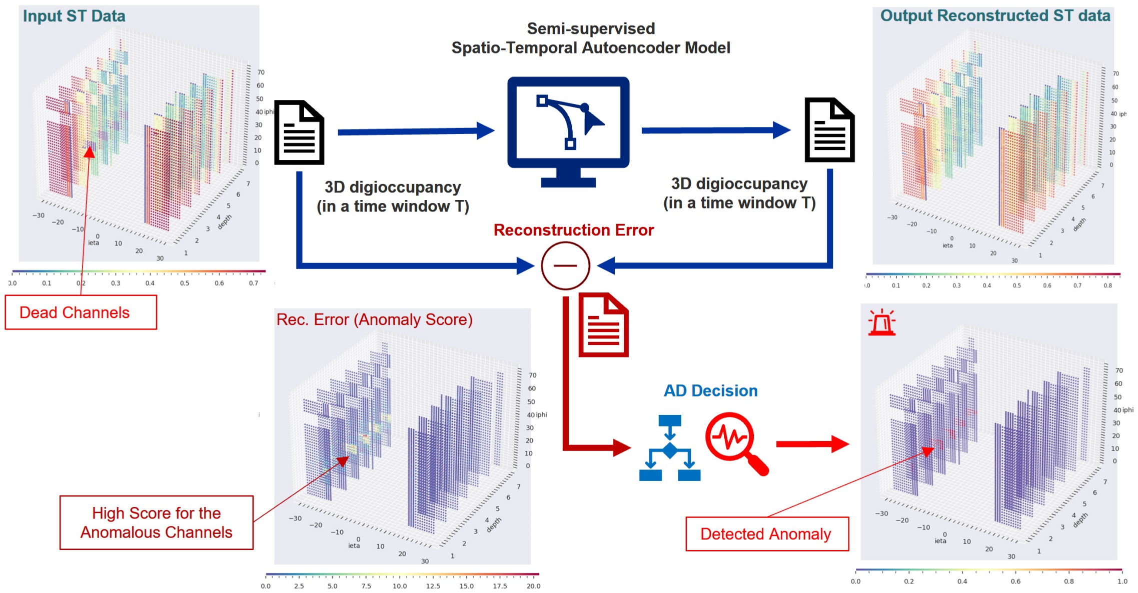


Figure 4.15: The proposed channel localized AE reconstruction AD system. The AE reconstructs the input ST digi-occupancy map, and spatial AD decision is performed using the anomaly scores estimated from the ST reconstruction errors.

γ_s is calculated as:

$$\gamma_s = \sum_{\forall i} \gamma(s, i) \tag{4.13}$$

where the $\gamma(s, i)$ is the digi-occupancy of the i^{th} channel in the map at the s^{th} LS. Finally, the per-channel $\gamma(s, i)$ is renormalized by its corresponding $\bar{\gamma}_s$ as:

$$\hat{\gamma}(s, i) = \frac{K\gamma(s, i)}{\bar{\gamma}_s} \tag{4.14}$$

where the $\hat{\gamma}$ is the renormalized γ, and the K is a scaling factor to compensate for the difference in the number of channels on the depth axes.

We employ fully-connected (FC) neural networks to build the regression model to effectively capture the non-linear relationships illustrated in Fig. 4.16:

$$input(\xi, \beta) \rightarrow \text{ReLU}(\text{FC}(64)) \rightarrow \text{ReLU}(\text{FC}(64)) \rightarrow \text{ReLU}(\text{FC}(7)) \rightarrow output(\bar{\gamma}_s) \tag{4.15}$$

Fig. 4.17 depicts data distribution of the γ_s before and after renormalization with $\mathcal{R}$. The renormalization has successfully handled the discrepancies on the γ_s from several runs—overlaps and centers distributions of $\hat{\gamma}_s$ and minimizes the outliers.

Adjacency Matrix Generation for Graph Network: We employ an undirected graph network $\mathcal{G}(\mathcal{V}, \Theta)$ to represent the calorimeter channels in a graph network based on their connection to a shared RBX system. The graph $\mathcal{G}$ contains nodes $\upsilon \in \mathcal{V}$, with edges $(\upsilon_i, \upsilon_j) \in \Theta$ in a binary adjacency matrix $\mathcal{A} \in \mathbb{R}^{M \times M}$, where M is the number of channel nodes. An edge indicates the channels sharing the same RBX as:

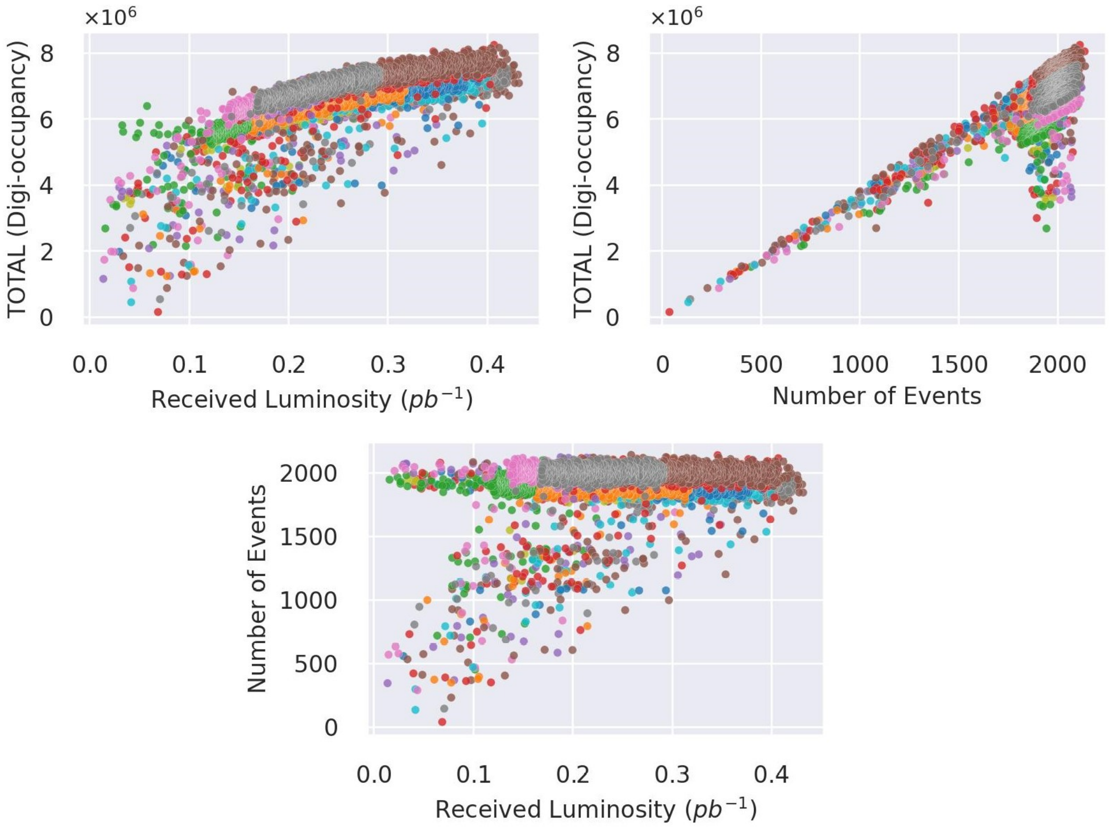


Figure 4.16: Digi-occupancy and run settings—the received luminosity and the number of events—in LS granularity. The number of events did not fully follow the drop in the luminosity (bottom plot) and the digi-occupancy (top-right plot); it portrays the non-linear behavior of LHC. The different colors correspond to different collision runs.

$$A(\upsilon_i, \upsilon_j) = \begin{cases} 1, & \text{if } \Omega(\upsilon_i) = \Omega(\upsilon_j) \\ 0, & \text{otherwise} \end{cases} \tag{4.16}$$

where $\Omega(\upsilon)$ returns the RBX ID of the channel υ. There are about 7K channels in a graph representing the digi-occupancy map of the HE calorimeter—each RBX network contains roughly 190 nodes. We retrieved the channel to RBX mapping from the HE segmentation map table (see Fig. 1.13).

Anomaly Detection Modeling

We denote the AE model of the GraphSTAD system as $\mathcal{F}$. The ST data is $\mathrm{X} \in \mathbb{R}^{T \times N_{i\eta} \times N_{i\phi} \times N_d \times N_f}$ as a sequence in a time window $t_x \in [t-T, t]$, where $N_{i\eta} \times N_{i\phi} \times N_d$ is the spatial dimension corresponding to the $i\eta$, $i\phi$, and *depth* axes, respectively, and $N_f = 1$ is the number of input variables—only a digi-occupancy quantity in the spatial data. The $\mathcal{F}_{\theta,\omega} : \mathrm{X} \rightarrow \bar{\mathrm{X}}$—parameterized by θ and ω—attempts to reconstruct the input ST data X and outputs $\bar{\mathrm{X}}$. The encoder network of the model $E_\theta : \mathrm{X} \rightarrow \mathbf{z}$ provides low-dimension latent space $\mathbf{z} = E_\theta(\mathrm{X})$, and the decoder

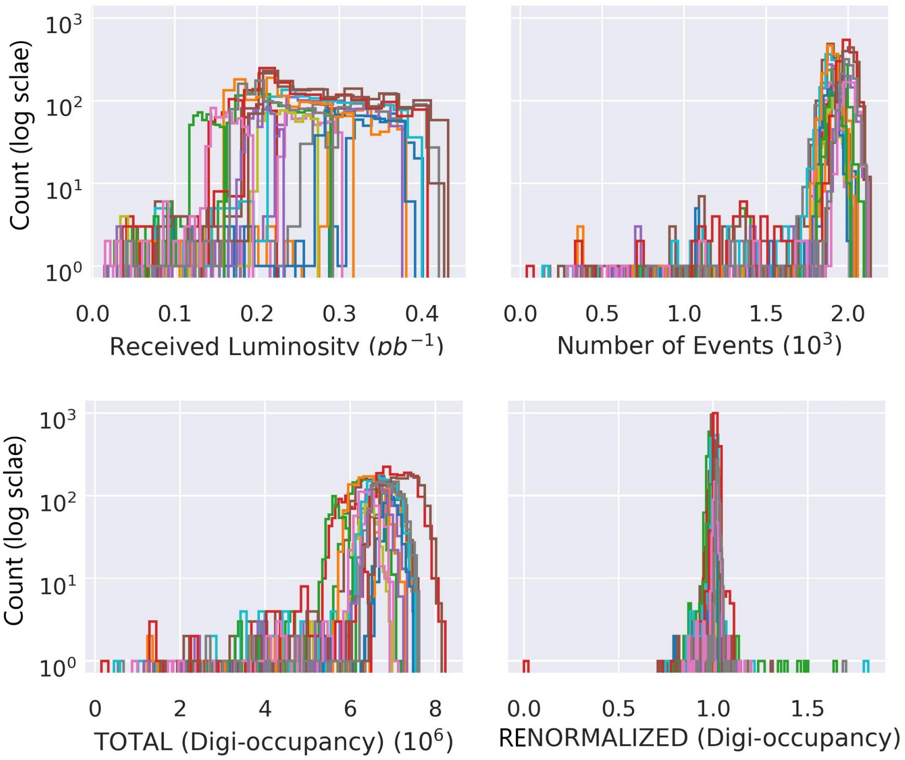


Figure 4.17: Distribution of total digi-occupancy per LS before and after renormalization. From left to right: (top) the received luminosity, and the number of events; (bottom) the digi-occupancy, and the renormalized digi-occupancy obtained with the regression model described in the text. The different colors correspond to different runs.

$D_\omega : \mathbf{z} \rightarrow \bar{\mathrm{X}}$ reconstructs the ST data from $\mathbf{z}$—$\bar{\mathrm{X}} = D_\omega(\mathbf{z})$ as:

$$\bar{\mathrm{X}} = \mathcal{F}_{\theta,\omega}(\mathrm{X}) = D_\omega(E_\theta(\mathrm{X})) \tag{4.17}$$

The channel anomalies can be transients—live for a short time and impact only a single digi-occupancy map—or persist over time—affecting a sequence of maps. The spatial reconstruction error e is calculated to detect a transient anomaly as:

$$e_i = |x_i - \bar{x}_i| \tag{4.18}$$

where $x_i \in \mathrm{X}$ and $\bar{x}_i \in \bar{\mathrm{X}}$ are the input and reconstructed digi-occupancy of the i^{th} channel. The e_i detects channel abnormality occurrence on isolated maps. We engage an aggregated error in a time window T using mean absolute error (MAE) to capture a time-persistent anomaly as:

$$e_{i,MAE} = \frac{1}{T} \sum_{t'=t-T}^{t} e_i(t') \tag{4.19}$$

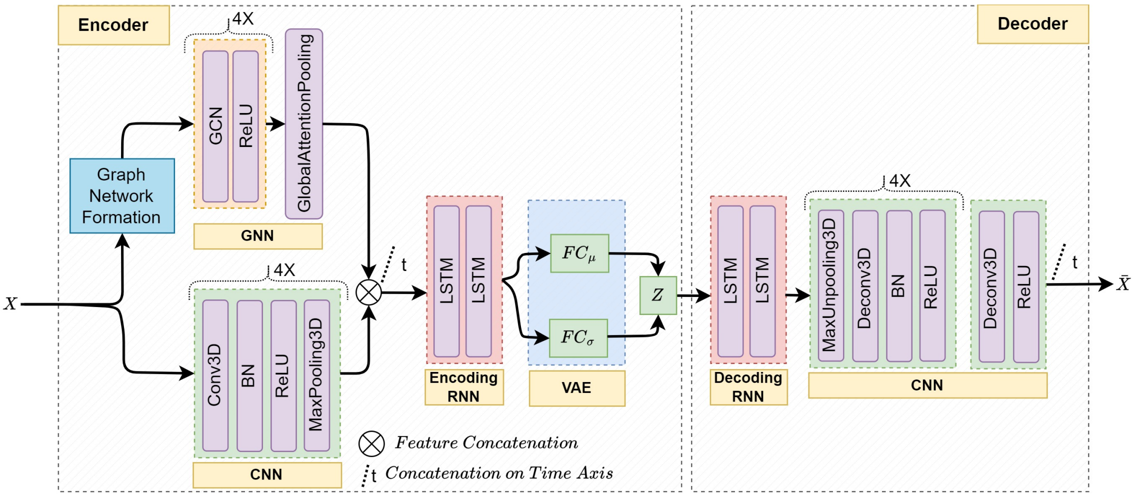
We standardize e_i to regularize the reconstruction accuracy variations among the channels—allowing a single AD decision threshold α to all the channels in the spatial map—as:

$$s_i = \frac{e_i}{\sigma_i} \tag{4.20}$$

where σ_i is the standard deviation of the e_i—or $e_{i,MAE}$ if the time window is considered—on the training dataset. The anomaly flags a_i are generated after applying α to the anomaly scores—$a_i = s_i > \alpha$. The α is a tunable constant that controls the detection sensitivity.

Model Architecture and Training

Convolutional neural networks have achieved state-of-the-art performance in several applications—mainly with image data [111, 117–120]. The shared nature of the kernel filters of the CNNs substantially reduces the number of trainable parameters in the model compared to fully-connected neural networks. Directly supplying the learned spatial features to temporal neural networks such as RNN could become inherently challenging due to the considerable computational demand for high-dimensional data. We employ CNN and GNN with a pooling mechanism to extract relevant features from high dimensional spatial data followed by RNN to capture temporal characteristics of the extracted features (see Fig. 4.18). We integrate variational layer [83] at the end of the encoder for regularization of AE overfitting by enforcing continuous and normally distributed latent representations [41, 97, 306, 352].



Conv3D: 3D convolutional neural network; GCN: graph convolutional neural networks; Deconv3D: 3D deconvolutional neural networks; BN: batch normalization; LSTM: long short-time memory recurrent networks; FC: fully-connected neural networks.

Figure 4.18: The architecture of the proposed AE for the GraphSTAD system. The GNN and CNN are spatial feature extraction on each time step, and the RNN network captures the temporal behavior of the extracted features. The feature extraction encoder incorporates the GNN for backend physical connectivity among the spatial channels, CNN for regional spatial proximity of the channels, and RNN for temporal behavior extraction. The reconstruction decoder contains RNN and deconvolutional neural networks to reconstruct the spatio-temporal input data from the low dimensional latent features.

The CNN of the encoder has L_c networks of Conv3D$(\cdot, kernel_size = [3 \times 3 \times$

3])[3] for regular spatial learning followed by batch normalization (BN)[4] for network weight regularization and faster convergence, ReLU for nonlinear activation, and MaxPooling3D[5] for spatial dimension reduction. The model can be summarized as:

$$y_t^c, \psi_t^c = \text{Pool}(\text{ReLU}(\text{BN}(\text{Conv3D}(x_t^l, N_c^l))))|_{l=1,...,L_c} \tag{4.21}$$

where x_t^l is the input spatial γ map data at time-step t and the N_c^l is the feature size of the l^{th} network. The y_t^c is the extracted feature set of the CNN at t. The Pool$(\cdot)$ denotes MaxPooling3D$(\cdot, stride = [2 \times 2 \times 2])$. The ψ_t^c holds the pooling spatial location indices of the MaxPooling3D layers to be used later for upsampling in the decoder during map reconstruction. The final extracted feature set $\text{Y}_c \in \mathbb{R}^{T \times N_c}$ of the CNN is an aggregation of all y_t^c in the time window T—concatenated on the time dimension—as:

$$\text{Y}_c = [y_1^c, y_2^c, ..., y_T^c] \tag{4.22}$$

We have used $L_c = 4$ to map the input spatial dimension $[64 \times 72 \times 7]$ into $[4 \times 4 \times 1]$, which yields a reduction factor of 2^{L_c} and expands the feature space of the input from $N_f = 1$ to $N_c = 128$. The $N_c' : [4 \times 4 \times 1 \times 128] = 2048$ spatial features are generated after reshaping.

The GNN of the encoder has L_g networks of a graph convolutional network (GCN)[6] with ReLU activation, and a final global attention pooling[6]. The networks are summarized as:

$$\begin{aligned} y_t^g &= \text{Pool}(\text{ReLU}(\text{GCN}(x_t^l, N_g^l)|_{l=1,...,L_g})) \\ \text{Y}_g &= [y_1^g, y_2^g, ..., y_T^g] \end{aligned} \tag{4.23}$$

where the GCN layers have a feature size of N_g^l, and the Pool$(\cdot)$ signifies the GlobalAttentionPooling$(\cdot)$ at the end of the GNN. The GlobalAttentionPooling aggregates the graph node features with an attention mechanism to obtain the final feature set of the GNN $\text{Y}_g \in \mathbb{R}^{T \times N_g}$. Similar to the CNN, we set $L_g = 4$ and $N_g = 128$ to generate the Y_g.

The encoded ST feature set $\zeta \in \mathbb{R}^{1 \times N_z}$ is obtained by learning the temporal context on the extracted spatial features $\text{Y} = [\text{Y}_c, \text{Y}_g]$ with two layers of long short-time memory (LSTM)[7] as:

$$\zeta = \text{LSTM}(\text{Y}, N_r^l)|_{l=1,2} \tag{4.24}$$

where N_r^l is the feature size of the l^{th} LSTM layer. The last layer $(N_r^2 = N_z = 32)$ generates the low-dimensional latent representation of the encoder. The VAE layer of the encoder generates the normally distributed representation latent features $\mathbf{z}$ as:

$$\mathbf{z} = \mu_z + \sigma_z \odot \epsilon \tag{4.25}$$

[3] https://pytorch.org/docs/stable/generated/torch.nn.Conv3d.html

[4] https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm3d.html

[5] https://pytorch.org/docs/stable/generated/torch.nn.MaxPool3d.html

[6] https://docs.dgl.ai/en/0.2.x/tutorials/models/1_gnn/1_gcn.html

[7] https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

where $\odot$ signifies an element-wise product with standard normal distribution sampling $\epsilon \sim \mathcal{N}(0, 1)$ [306]. The μ_z and the σ_z of the VAE are implemented with FC[8] layers taking the ζ as input.

The decoder network of the AE is made of RNN and CNN to reconstruct the target ST data from the latent features. The decoding embarks with temporal feature reconstruction using LSTM network as:

$$\bar{\zeta} = \text{LSTM}(\mathbf{z}, N_r^l)|_{l=1,2} \tag{4.26}$$

where $\bar{\zeta}$ is the reconstructed temporal feature set from the latent space $\mathbf{z}$. Spatial reconstruction follows for each time-step t through a multi-layer deconvolutional neural network. Each network starts with MaxUnpooling3D$(\cdot, \mathit{stride} = [2 \times 2 \times 2], \psi_l^c)$[9] to upsample the spatial data using localization indices ψ_l^c from the l^{th} MaxPooling3D of the encoder followed by a deconvolutional layer (Deconv3D$(\cdot, \mathit{kernel_size} = [3 \times 3 \times 3])$) [355], BN and ReLU. Eventually, Deconv3D$(\cdot, \mathit{kernel_size} = [1 \times 1 \times 1])$ is incorporated for final output stabilization. The decoder network is summarized as:

$$\begin{aligned} \bar{x}_t &= \text{ReLU}(\text{BN}(\text{Deconv3D}(\text{Unpool}(\bar{\zeta}_t, \psi_t^c), N_c^l)))|_{l=1,\ldots,L_c} \\ \bar{x}_t &= \text{ReLU}(\text{Deconv3D}(\bar{x}_t, N_f)) \end{aligned} \tag{4.27}$$

where the $\bar{x}_t$ is the reconstructed spatial data, and the Unpool$(\cdot)$ denotes MaxUnpool3D$(\cdot)$. The final reconstructed ST data $\bar{\mathrm{X}} \in \mathbb{R}^{T \times N_{i\eta} \times N_{i\phi} \times N_d \times N_f}$ is obtained as:

$$\bar{\mathrm{X}} = [\bar{x}_1, \bar{x}_2, ..., \bar{x}_T] \tag{4.28}$$

We trained the AE on healthy digi-occupancy maps of LHC collision runs. The modeling task becomes a multivariate learning problem since the target data contains readings from multiple calorimeter channels in the spatial digi-occupancy map. Appropriate scaling of the spatial data is thus necessary for effective model training; we further normalized the spatial data per channel into a range of $[0, 1]$. We have also observed that the γ distribution of the channels at the first depth of the spatial map is different from the channels at the higher depths (see Fig. 4.14); distribution imbalance on target channel data affects model training efficacy when well-known statistical algorithm—e.g., MSE—is employed as loss functions. MSE loss minimizes the cost of the entire space, and it may converge to a non-optimal local minimum in the presence of imbalanced data distribution; this phenomenon is known as the class imbalance challenge in machine learning classification problems. A widely used remedy is to employ a weighting mechanism—assigning weights to the different targets. We applied a weighted MSE loss function to scale the loss from the different distributions—the $\mathit{depth} \in 1$ and $\mathit{depth} \in 2, \ldots, 7$—as:

$$\mathcal{L}' = \sum_j \frac{\varsigma_j}{M_j} \sum_{i \in C_j} (x_i - \bar{x}_i)^2 \tag{4.29}$$

[8] https://pytorch.org/docs/stable/generated/torch.nn.Linear.html

[9] https://pytorch.org/docs/stable/generated/torch.nn.MaxUnpool3d.html

where x_i is the $\hat{\gamma}$ of the i^{th} channel in the j^{th} group set C_j, M_j is the number of channels in C_j, and ς_j is the weight factor of the MSE loss of the j^{th} group. We holistically set $\varsigma_1 = 0.4$ and $\varsigma_2 = 1$ after experimenting with different ς values.

The VAE regularizes the training MSE loss using the KL divergence loss D_{KL} to achieve the normally distributed latent space as:

$$\mathcal{L} = \underset{W \in \mathbb{R}}{\operatorname{argmin}} \left\{ \mathcal{L}' - \beta D_{KL} \left[\mathcal{N}(\mu_z, \sigma_z), \mathcal{N}(0, I) \right] + \rho \|W\|_2^2 \right\} \tag{4.30}$$

where $\mathcal{N}$ is a normal distribution with zero mean and unit variance, and $\|.\|_2^2$ is the squared *Frobenius norm* of L_2 regularization for the trainable model parameters W. The $\beta = 0.003$ and $\rho = 10^{-7}$ are tunable regularization hyperparameters. We used *Adam* optimizer with super-convergence *one-cyclic* learning rate scheduling [356] for training.

4.2.3 Experimental Results and Discussion

AD studies for the DQM inject simulated anomalies into good data to validate the effectiveness of the developed models since a small fraction of the data is affected by real anomalies [39]. We trained the AE model using four GPUs on 10K digi-occupancy maps—from LS sequence number $[1, 500]$—and evaluated on LSs $[500, 1500]$ injected with synthetic anomalies simulating real dead, hot, and degraded calorimeter channels. We employed early-stopping using 20% of the training dataset to estimate the validation loss during each training epoch (see Fig. 4.19). The model training achieved good fitting and generalization, as demonstrated by the low loss and closeness between the training and validation losses.

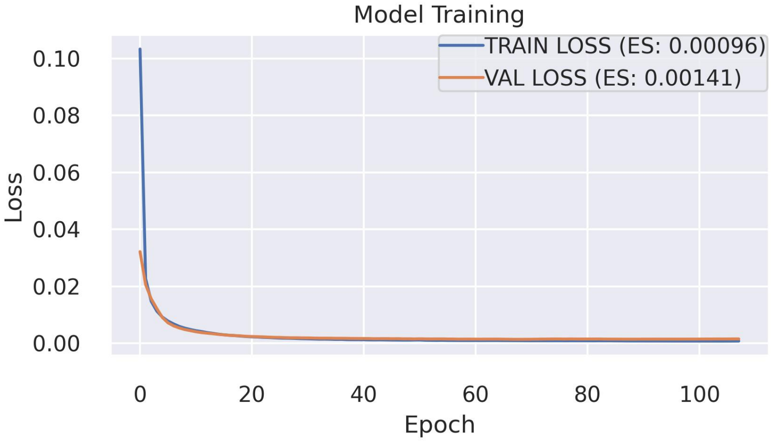


Figure 4.19: GraphSTAD AE model training (*early-stopping* = 20 *epochs, learning rate* = 10^{-3}, *weight regularization* = 10^{-7}, *training time* = 82 *minutes*). The low training loss indicates good model fitting—no under-fitting—to the data set, and the low validation loss demonstrates good generalization—no over-fitting.

Fig. 4.20 demonstrates the capability of the proposed ST AE in reconstructing normal digi-occupancy maps from a sequence of lumisections. The AE has accomplished a promising reconstruction ability on the ST digi-occupancy data. High reconstruction accuracy on the healthy data is essential to reduce false-positive flags when a semi-supervised AE is employed for AD application. We will further discuss the reconstruction error distribution comparison on the healthy and abnormal channels in the AD performance section.

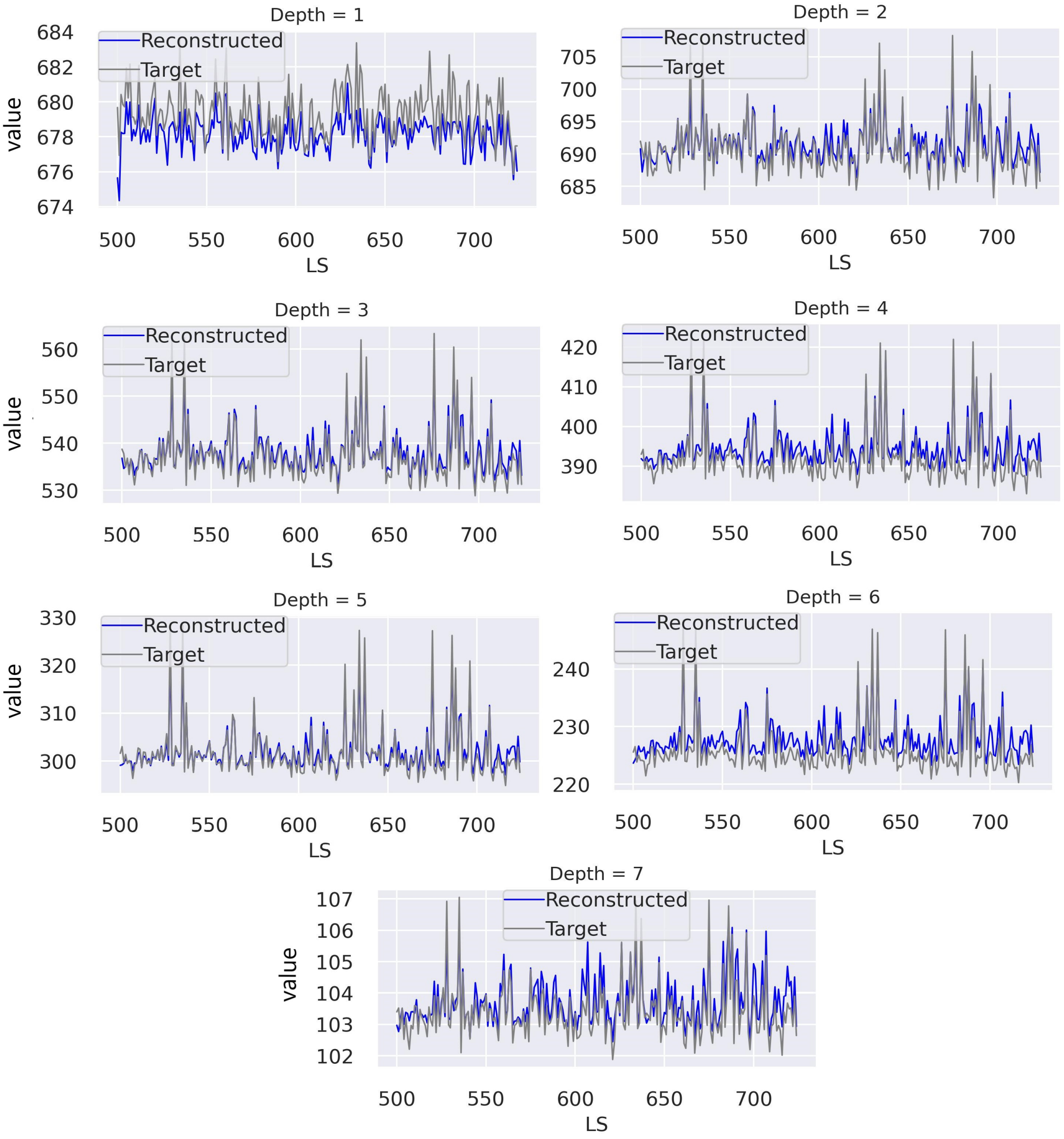


Figure 4.20: ST digi-occupancy maps reconstruction on samples from the test dataset (*RunId: 325170, LS=[500, 750]*). The figure illustrates the total digi-occupancy across the seven depths—$\hat{\gamma}_l$. Our GraphSTAD AE operates on ST γ map data, and we present the above plots—corresponding to the γ_l per LS—to demonstrate capability of the AE in handling fluctuation across a sequence of LS.

We created synthetic anomalies to simulate dead, hot, and degraded channels and then injected them into healthy digi-occupancy maps. We subsequently evaluated the ability of the AD to detect the injected anomalies. The anomaly generation algorithm involves three steps: 1) selection of a random set of LSs $\tau \in [500, 1500]$

from the test set, 2) random selection of spatial locations φ for each τ, where $\varphi \in \{i\eta \times i\phi \times depth\}$ on the HE axes (see Fig. 4.14), and 3) injection of anomalies. The anomalies are simulated using degrading factor R_D that $\gamma_a = R_D\gamma_h$, where γ_a and γ_a are the healthy and anomaly channel γ values: the dead ($R_D = 0$, and $\gamma_a = 0$), hot ($R_D > 1$, and $\gamma_a >> \gamma_h$), and degraded ($0 < R_D < 1$, and $0 \leq \gamma_a < \gamma_h$). We have kept the same τ and φ of the generated anomalies for consistency when evaluating the AD performance of the different anomaly types.

Detection of Dead and Hot Channels: We have evaluated the AD accuracy on dead—$\gamma_a = 0,\ R_D = 0$—and hot—$\gamma_a = R_D\gamma_h,\ R_D = 200\%$—channels on the 10K maps—5K maps for each anomaly type. Table 4.3 and Table 4.4 present the AD performance on transient anomalies—short-lived in isolated maps—and time persisting anomalies—encroach consecutive maps in a time window, respectively. Our model achieves good accuracy with precise localization of the faulty channels—0.99 precision when capturing 99% of the 335K faulty channels. Time-persistent anomalies are easier to detect—the FPR generally improves by 13%-23% and 28%-40% for the dead and hot anomalies, respectively, compared to the short-lived anomalies on isolated LSs. We have observed that most false positives (FPs) occur on channels with low expected γ_h, where the model achieves relatively lower reconstruction accuracy. The performance is not entirely unexpected since we trained the AE to minimize a global MSE loss function (4.30). The reconstruction errors become relatively high for channels with low γ ranges that limit effectiveness in distinguishing the anomalies when capturing 99% of the time-persistent dead channels using (4.19).

We have monitored roughly 31.28M HE sensor channels—of which 335K (1.07%) are simulated abnormal channels—from the 5K maps on the isolated map evaluation in Table 4.3. The monitored channels grow to 156M with 1.68M (1.07%) anomalies for the evaluation of time-persistent anomalies in Table 4.4 using time window five maps resulting in 25K maps.

Table 4.3: AD on dead and hot channel anomalies on isolated digi-occupancy maps.

Anomaly Type	**Captured Anomalies**	**P**	**R**	**F_1**	**FPR**
Dead Channel ($R_D = 0\%$)	99%	0.999	0.99	0.995	6.722×10^{-6}
	95%	1.000	0.95	0.974	3.102×10^{-6}
	90%	1.000	0.90	0.947	2.068×10^{-6}
Hot Channel ($R_D = 200\%$)	99%	0.999	0.99	0.994	9.113×10^{-6}
	95%	1.000	0.95	0.974	1.939×10^{-6}
	90%	1.000	0.90	0.947	1.196×10^{-6}

* P- precision, R- recall, F_1- f1-score, and FPR- false positive rate

Detection of Degrading Channels: Table 4.5 presents the AD accuracy of time-persistent degraded channels simulated with $R_D = [80\%, 60\%, 40\%, 20\%, 0\%]$; the $R_D = 0\%$ corresponds to a dead channel. We injected the generated channel faults into 1K maps for each decay factor. We have monitored around 156M channels—of which 1.74M (1.11%) are abnormal channels—in the total of 25K digi-occupancy

Table 4.4: AD on time-persistent dead and hot channel anomalies.

Anomaly Type	**Captured Anomalies**	**P**	**R**	**F$_1$**	**FPR**
Dead Channel ($R_D = 0\%$)	99%	0.999	0.99	0.995	7.691×10^{-6}
	95%	1.000	0.95	0.974	2.715×10^{-6}
	90%	1.000	0.90	0.947	1.616×10^{-6}
Hot Channel ($R_D = 200\%$)	99%	0.999	0.99	0.995	5.461×10^{-6}
	95%	1.000	0.95	0.974	1.357×10^{-6}
	90%	1.000	0.90	0.947	7.756×10^{-7}

maps—5K maps per in the time window. The AD system has demonstrated promising potential in detecting degraded channel anomalies. The FPR to capture 99% of the anomaly is 2.988%, 0.155%, 0.022%, 0.002%, and 0.001% when channels operate at 80%, 60%, 40%, 20%, and 0% of their expected capacity, respectively.

The relatively lower precision at the $R_D = 80\%$ indicates that there are still a few anomalies challenging to catch despite the very low FPR considering the accurate classification of numerous true negative healthy channels (see Fig. 4.21); the channels operating at $R_D = 80\%$ are mostly inliers—overlapping with the healthy operating ranges—and detecting them is difficult when the expected γ_h of the channel is low. The significant improvement of the FPR by 88% and 95% when the amount of the captured anomaly is reduced to 95% and 90%, respectively, demonstrates a small percentage of the channels causes the performance drop at $R_D = 80\%$. Fig. 4.22 illustrates the overlap regions on the distribution of the reconstruction errors of the healthy and faulty channels at the various R_D values.

Table 4.5: AD on time-persistent degraded channels.

Anomaly Type	R_D	**FPR (90%)**	**FPR (95%)**	**FPR (99%)**
Degraded Channel	80%	1.636×10^{-3}	3.614×10^{-3}	2.988×10^{-2}
	60%	1.329×10^{-4}	3.834×10^{-4}	1.550×10^{-3}
	40%	8.405×10^{-6}	2.764×10^{-5}	2.242×10^{-4}
	20%	2.263×10^{-6}	5.173×10^{-6}	2.505×10^{-5}
	0%	9.699×10^{-7}	1.778×10^{-6}	6.142×10^{-6}

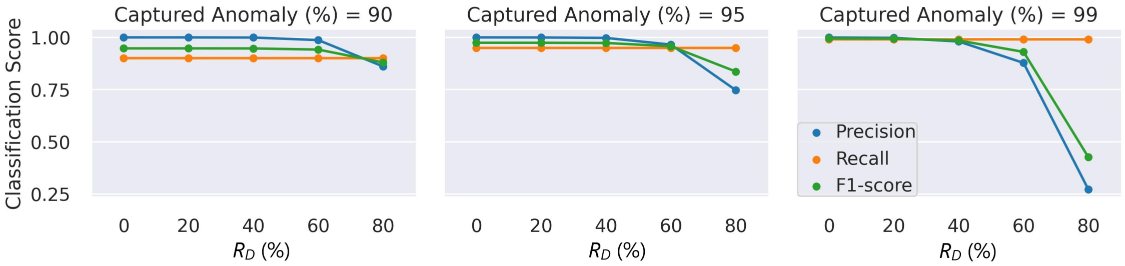


Figure 4.21: AD classification performance on time-persistent degraded channels.

We have quantitatively compared alternative benchmark models to validate the capability of the GraphSTAD (see Fig. 4.23). The benchmark AE models employ a similar architecture as the GraphSTAD AE but with different layers. The results

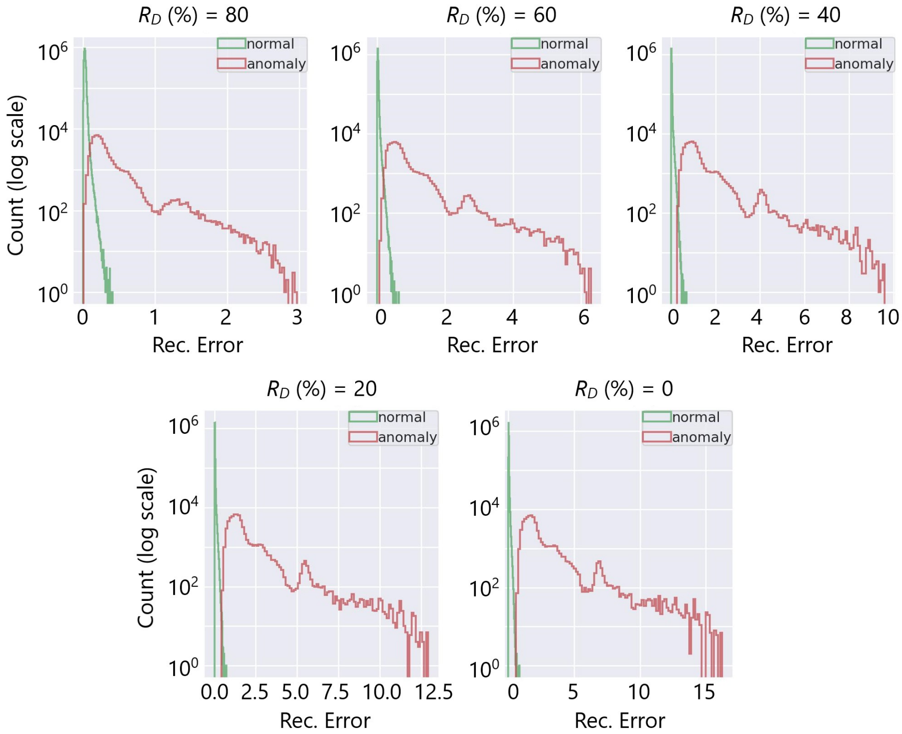
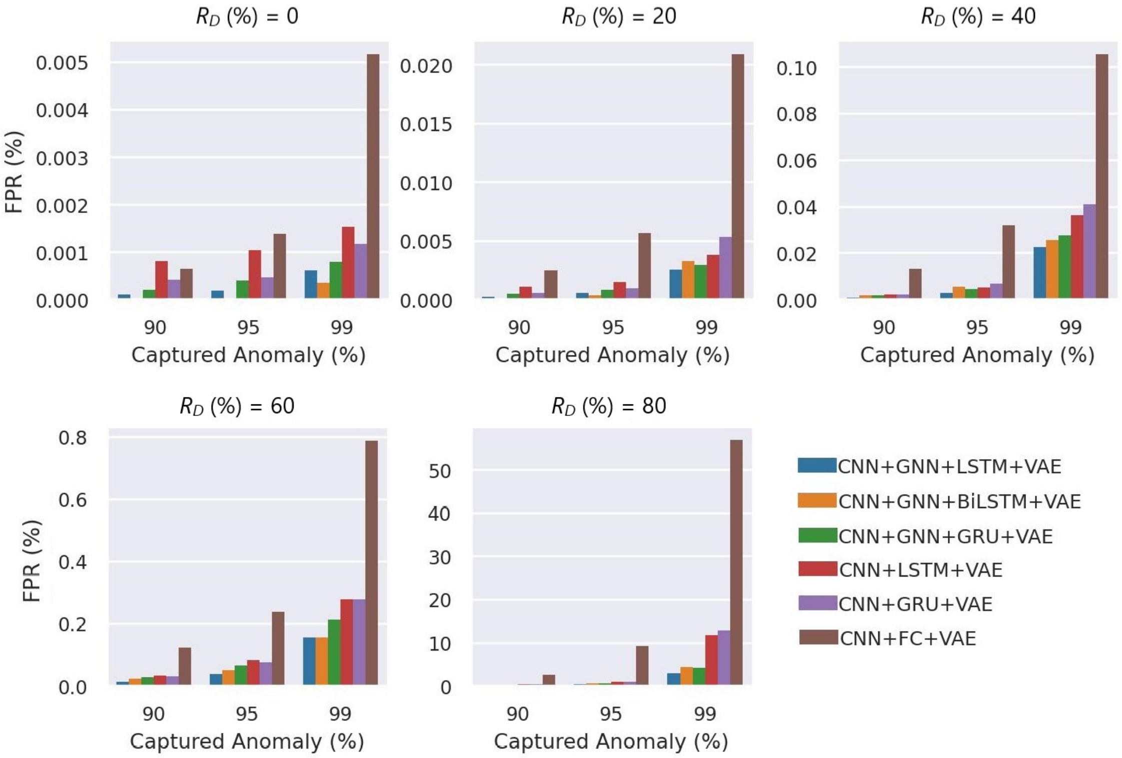


Figure 4.22: Reconstruction error distribution of healthy and anomalous channels at different R_D. The overlap region decreases substantially as the channel deterioration increases (left to right).

demonstrate that the integration of the GNN has a significant performance improvement from 1.6 to 3.9 times in the FPR. The temporal models with RNN have achieved a 3 to 5-fold boost over the non-temporal spatial AD model when capturing severely degraded channels. The GraphSTAD has a substantial 25 times amelioration over the non-temporal model for subtle and inlier anomalies—e.g., channels deteriorate by 20% at $R_D = 80\%$. Incorporating temporal modeling and GNN has enhanced degrading channel detection performance.

Detection of Real Anomalies in the HCAL: Our GraphSTAD system has caught five real faulty HE channels in collision data *RunId=324841* using the digi-occupancy maps. The faulty channels are located at $[i\eta = 17, i\phi = 71, depth = 3]$, $[i\eta = 18, i\phi = 71, depth = 3]$, $[i\eta = 18, i\phi = 71, depth = 4]$, $[i\eta = 18, i\phi = 71, depth = 5]$, and $[i\eta = 28, i\phi = 71, depth = 4]$, and have impacted 52 consecutive LSs (see Fig. 4.24). Fig. 4.24 and Fig. 4.25 illustrate the detected faults fall into the dead channel category except in the last *LS=57* where the channels operated in a degraded state—the γ is lower than expected. Detecting degraded channels is challenging since the γ reading is non-extreme like in dead and hot channels, and the γ drop overlaps with other false down-spikes (see $LS > 57$ in Fig. 4.24). The down-spikes in the digi-occupancy for $LS > 57$ are due to non-linearity in the LHC—changes in collision run settings (see Fig. 4.24b); our normalizing regression model has successfully handled the fluctuation during prepossessing before causing

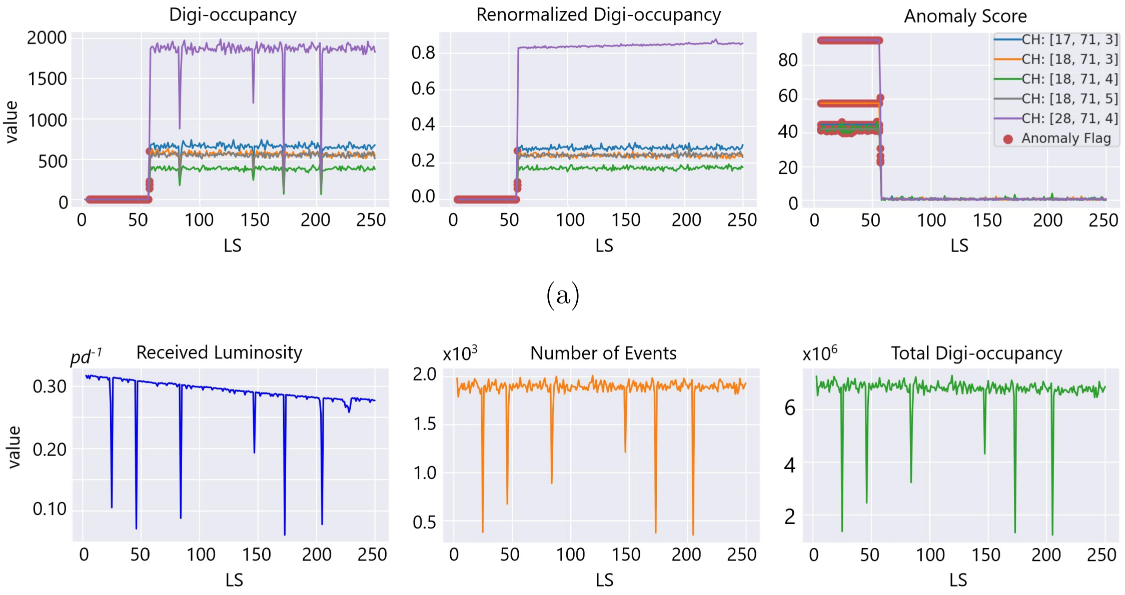


CNN: convolutional neural networks, GNN: graph neural networks, BiLSTM: bidirectional LSTM, GRU: gated recurrent unit, and VAE: variational AE.

Figure 4.23: Comparison with benchmark models on time-persistent anomaly channels. The GraphSTAD (CNN+GNN+LSTM+VAE) achieves a significantly lower FPR.

false-positive alerts (see Fig. 4.24a). Fig. 4.26 and Fig. 4.27 portray the spatial anomaly scores during death and degraded status of the faulty channels; the high anomaly scores localized at the faulty channels demonstrate the GraphSTAD AD performance at a channel-level granularity. The existing DQM system in CMS—uses rule-based and statistical methods—has also reported these abnormal channels at run-level analysis; the results are only available at the end of the run after analyzing all the LSs for the run [71]. Our approach is adaptive to variability in the digi-occupancy maps and provides anomaly localization that detects faulty—including non-extreme degraded—channels per lumisection granularity.

Cost Model Complexity: We developed the models with PyTorch and trained them on four GPUs of NVIDIA Tesla V100 SXM3 32GB and Intel(R) Xeon(R) Platinum 8168 CPU 2.70GHz. We utilized a time window $T = 5$ and batch size $B = 8$ for training, and the dimension of a batch is $[B \times T \times N_{i\eta} \times N_{i\phi} \times N_d \times N_f]$. The training time of the GraphSTAD model is approximately 45 seconds per epoch. The training iteration epoch 200 achieves good accuracy with a one-cycle learning rate schedule [356]. The non-temporal model—CNN+FC+VAE—is the fastest, and its leverage emanates from its non-recurrent networks that only analyze a single map instead of sequential processing of five maps in a time window. The median



(a)

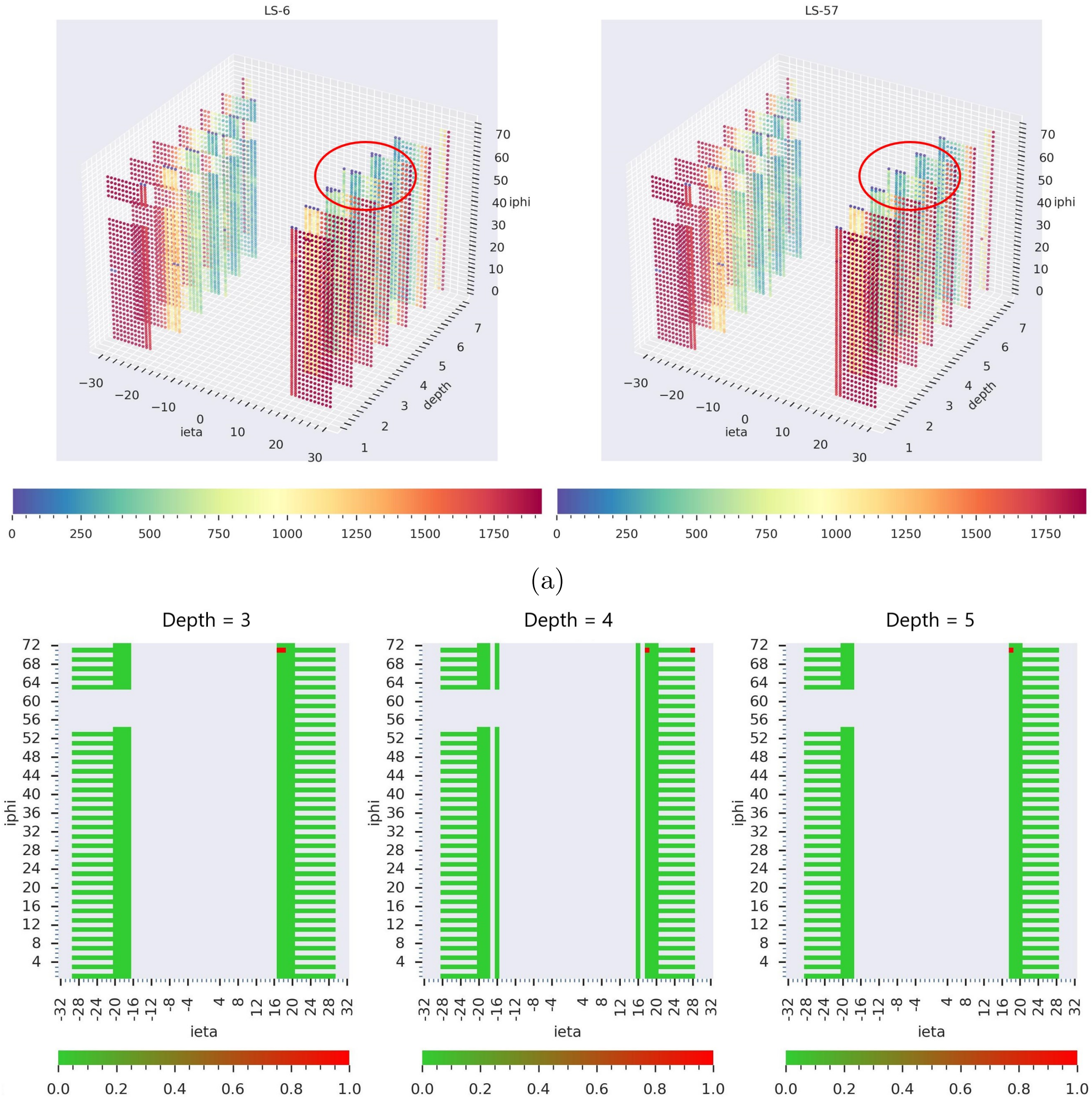
(b)

Figure 4.24: Detected real faulty channels on digi-occupancy maps at *LS=[6, 57]* of *RunId=324841*: a) the digi-occupancy dropped to near zero for the faulty channels (left and middle plots)—resulting in high anomaly scores (right), and b) collision run settings and the total digi-occupancy per LS. Dead (*LS=[6, 56]*) and degraded channel anomalies (*LS=57*) were captured on the highlighted LSs (red) in (a).

inference time of the GraphSTAD on a single GPU is roughly 0.05 seconds with a standard deviation of 0.006 seconds. The integration of the GNN makes the inference relatively slower compared to the benchmark models (see Fig. 4.28). The processing cost is within an acceptable range for CMS production requirements since the input digi-occupancy map is generated at each lumisection with a time interval of 23 seconds.

4.2.4 Summary

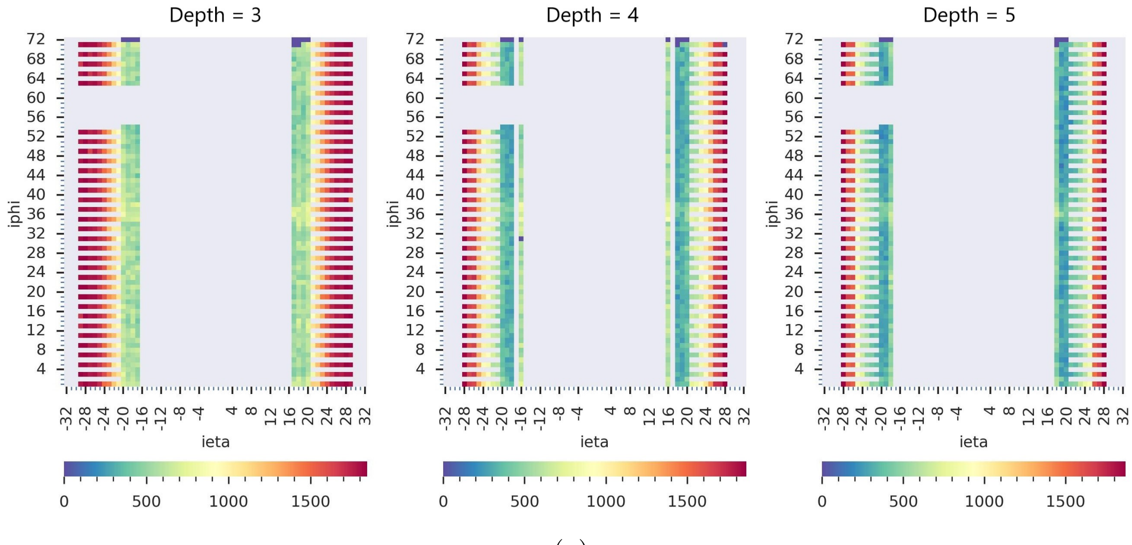
Our study has presented a semi-supervised anomaly detection system for the data quality monitoring system of the Hadron Calorimeter using spatio-temporal digi-occupancy maps. We have extended the synergy of temporal deep learning developments for the CMS experiment. Our approach has addressed modeling challenges—including digi-occupancy map renormalization, learning non-Euclidean spatial behavior, and degrading channel detection. We have proposed the GraphSTAD system that combines convolutional, graph, and temporal learning networks to capture spatio-temporal behavior and achieve robust localization of anomalies at a channel granularity on high spatial data. The AD performance evaluation has demonstrated the efficacy of the proposed system for channel monitoring. Our proposed AD system will facilitate monitoring and diagnostics of faults in the frontend particle hit sensing hardware and software system of the calorimeter. It will enhance the ac-



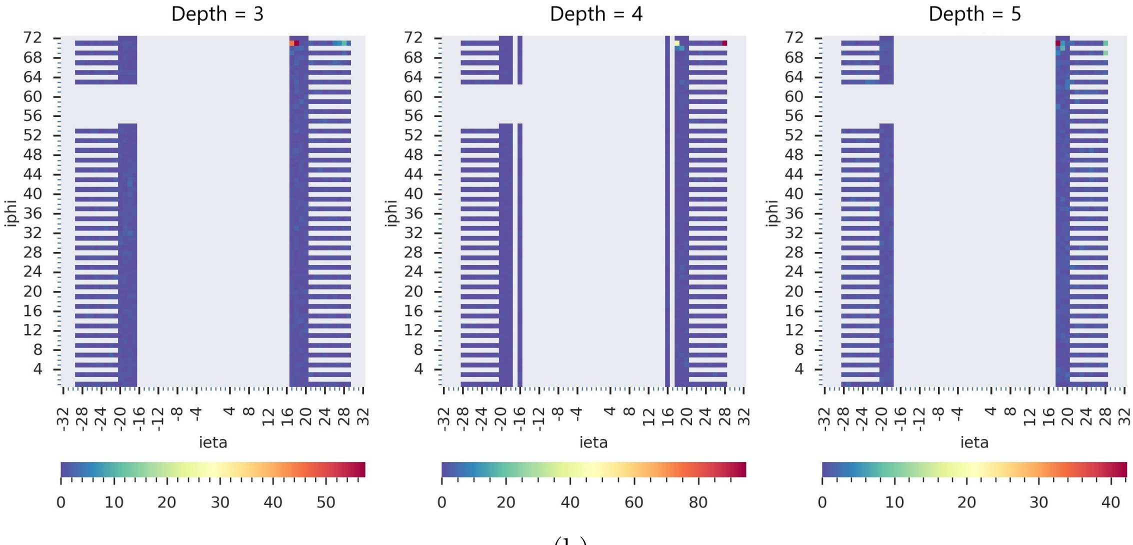
(a)

(b)

Figure 4.25: Spatial view on real faulty channels detection from *RunId=324841* collision run data: a) the 3D digi-occupancy maps with faulty dead channels on the left at *LS=6* and degraded channels on the right at *LS=57*, and b) the channel anomaly flags on the 2D map per the depth axes—red for anomaly and green for healthy. Previously known bad channels during model training are excluded in the plots and are not detected as new.

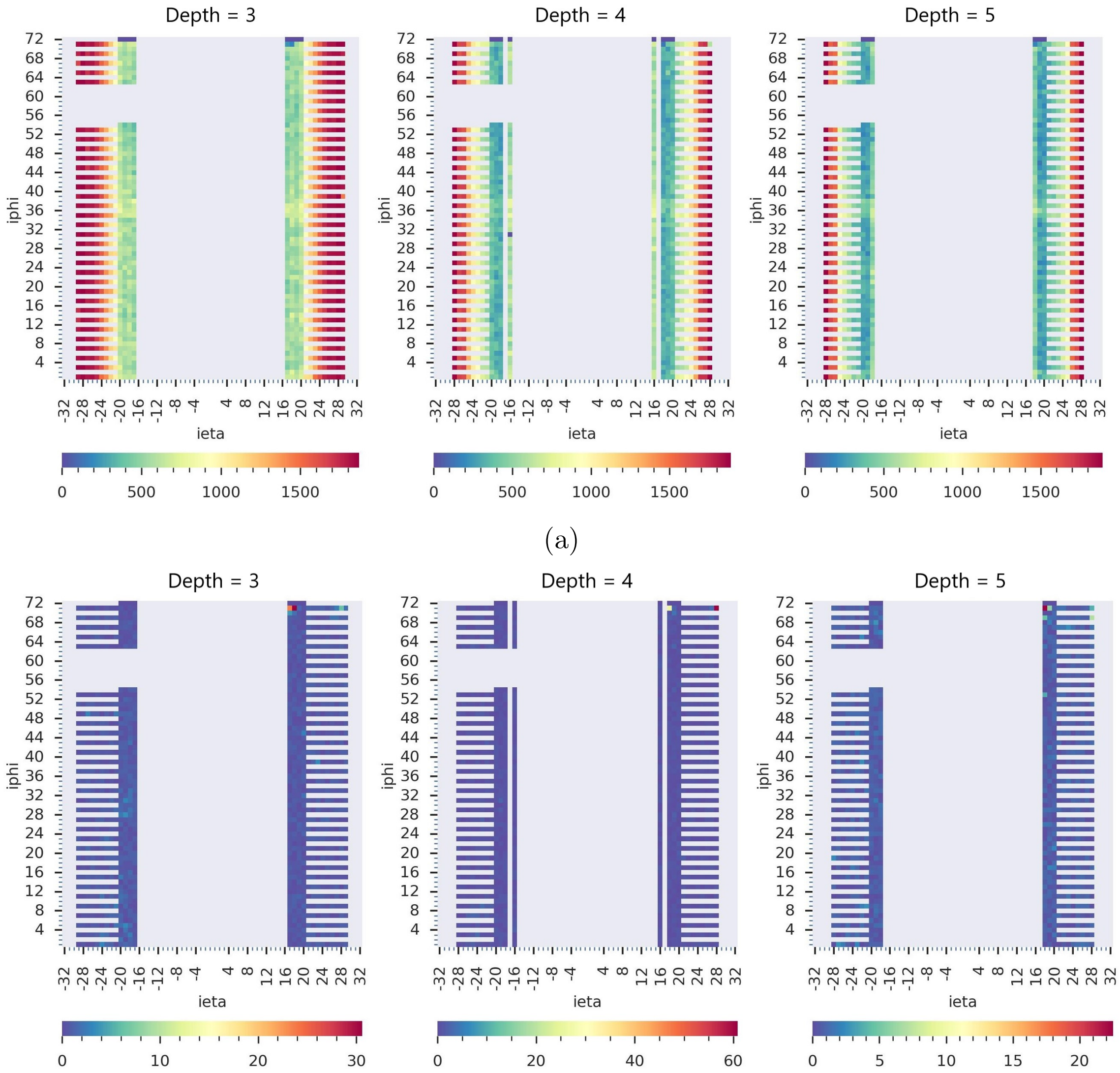


(a)



(b)

Figure 4.26: Spatial view on the detected real dead channels at the *LS=6* from the *RunId=324841*: a) the 2D digi-occupancy maps at the *depth* axes of the faulty channels, and b) the corresponding anomaly score maps. The GraphSTAD localizes the anomaly scores on the faulty dead channels.



(a)

(b)

Figure 4.27: Spatial view on the detected real degraded channels at the *LS=57* from the *RunId=324841*: a) the 2D digi-occupancy maps at the *depth* axes of the faulty channels, and b) the corresponding anomaly score maps. The GraphSTAD localizes the anomaly scores on the faulty degraded channels with strength proportional to anomaly severity—lower scores in the color bars than the dead channels.

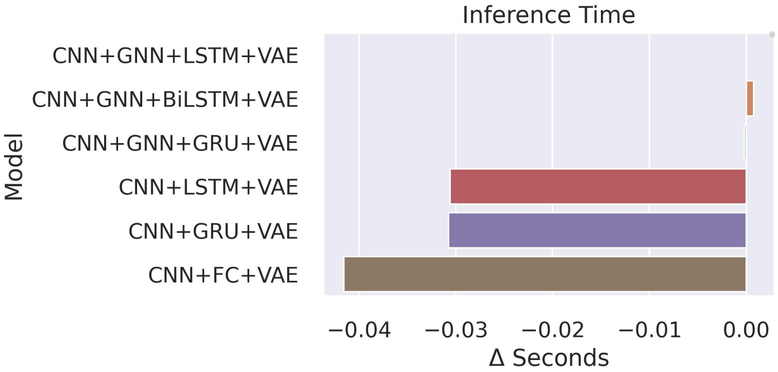


Figure 4.28: Model inference computational cost relative to the proposed GraphSTAD model (CNN+GNN+LSTM+VAE). The GNN increases the inference delay, whereas the non-temporal model (CNN+FC+VAE) has the speed advantage due to its relatively lower number of model parameters and inference on a single map instead of time windowing.

curacy and automation of the existing DQM system—providing instant anomaly alerts on a broader range of channel faults in realtime; the improved monitoring of the calorimeter will result in the collection of high-quality physics data. The methods and approaches discussed in this study are domain-agnostic and can be adopted in other spatio-temporal fields—particularly when the data exhibits regular and irregular spatial characteristics.

4.3 Extending Anomaly Detection for the Data Quality Monitoring through Transfer Learning between Calorimeters

The proliferation of sensors for various purposes—including monitoring, diagnostics, and prognostics of infrastructure—brings with it a large amount of data in many domains. The sheer data volume makes data curation time-consuming, and deploying data analytics platforms in new environments thus becomes expensive. Transfer learning (TL) utilizes previously trained models for new tasks and can mitigate the lack of curated data and model complexity. We present in this study both the potential benefits and limitations of TL within the context of spatio-temporal (ST) anomaly detection for the HCAL of the CMS experiment at CERN. We transfer the AD model trained on data collected from one calorimeter of the HCAL to another. We investigate different configurations of TL on the GraphSTAD model discussed in Section 4.2—transferring the convolutional, graphs, and recurrent neural networks. The experiment results demonstrate that TL can effectively extract and reconstruct ST features with and without fine-tuning on a new target data set. The TL achieves promising ST reconstruction and AD performance while substantially reducing the trainable parameters of the AD models. It also improves robustness against anomaly contamination in the training data sets of the semi-supervised AD models.

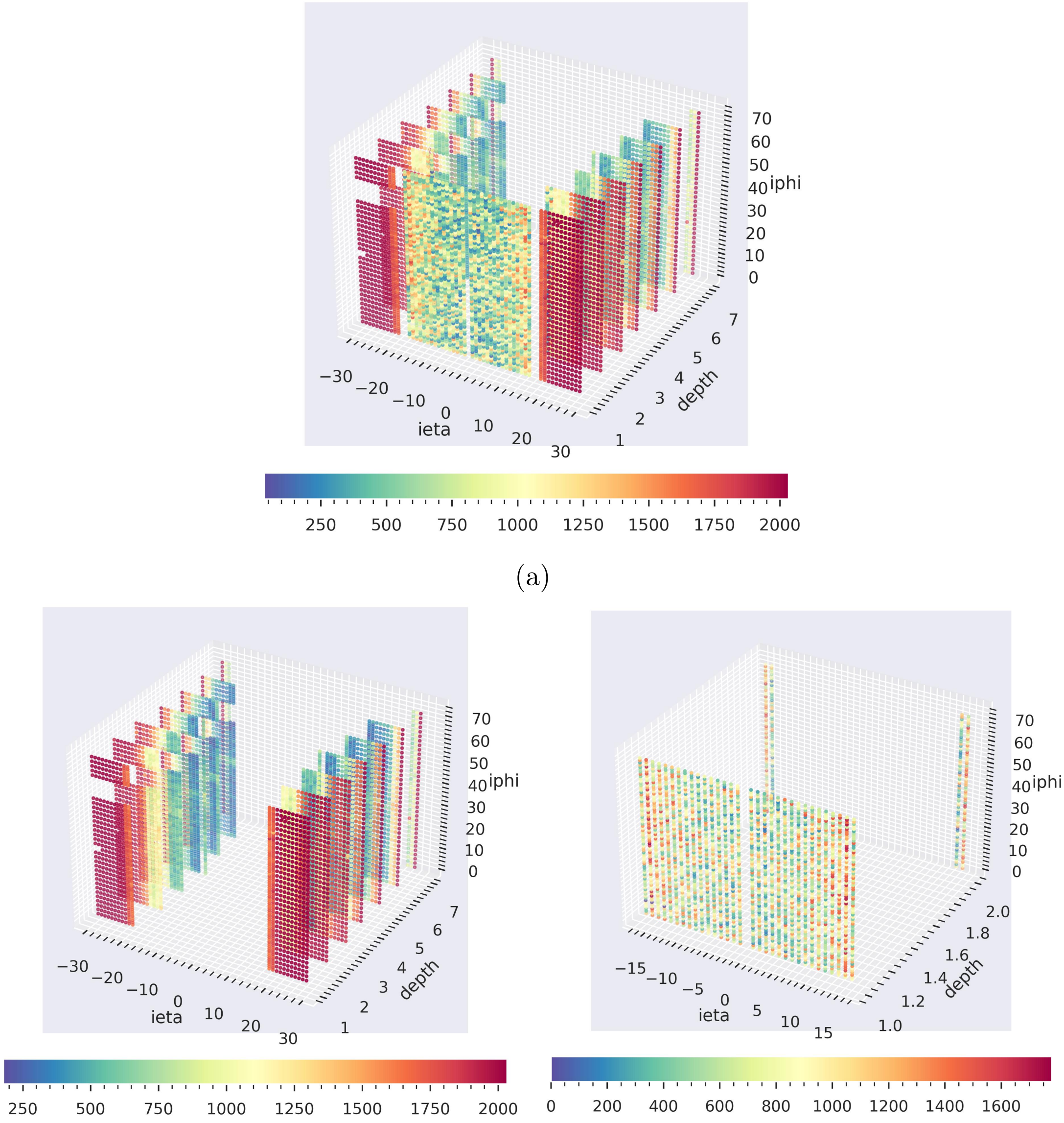
Semi-supervised AD models have accomplished promising performance in reliability, safety, and health monitoring applications in several domains [80–82]. The deployment of these AD models in a new environment is often circumscribed by the limited amount of clean data [79]. Data curation for ML modeling remains cumbersome and particularly challenging for temporal data despite data abundance. Transfer learning mechanisms have been proposed for DL models to mitigate the challenge of data insufficiency; it accelerates model training and enhances accuracy [79, 187–190, 194]. The aim is to achieve in-domain and cross-domain learning by extracting useful information from the model or data of a source task and transferring it to a target task [191, 193]. TL is widely employed in computer vision [191] and natural language processing [193]; it has also been investigated in TS sensor data for machine monitoring [187], electricity loads [188], medical [189], dynamic systems [190], and ST data for crowd prediction [79, 194]. The TL on ST data for AD application remains limited [193,197]. Further study on TL for ST AD models—often involving combinations of spatial and temporal learning networks—is essential considering the achievements of TL in other domains [193]. The most recent hybrid DL models are commonly made of combinations of two or more variants of CNNs, RNNs, GNNs, and transformers for various ST data mining tasks [79, 194, 197, 198]. Our study investigates TL on ST semi-supervised AD AE models.

We discuss the GraphSTAD system—an AE model built on CNN, RNN, and GNN— [77] to investigate TL for the ST AD task on the HCAL DQM. The GraphSTAD has been proposed for particle acquisition channel monitoring of the HCAL (see Section 4.2) [77]. It captures abnormal events using spatial appearance and temporal context on ST digi-occupancy maps of the DQM. The GraphSTAD model is designed to use CNNs to capture the behavior of adjacent channels exposed to regional collision particle hits, GNNs to learn local electrical and environmental characteristics due to a shared backend circuit of the channels, and RNNs to detect temporal degradation on faulty channels over time. We transfer the GraphSTAD AE—pre-trained on the source HCAL Endcap (HE) subsystem—into a different target subsystem—the HCAL Barrel (HB)—for the TL experiment. The HE and HB are subdetectors of the HCAL; they are designed to capture hadron particles at different positions of the calorimeter. The subdetectors share similarities but also have differences in design, technology, and detector segmentations [8]. We provide insights on TL using various training experiments with different network hierarchies of the GraphSTAD AE. The experiment demonstrates the potential of TL on the feature extraction—the encoder—and reconstruction—the decoder—networks with and without fine-tuning on the target dataset. We also examine the impact of within and across time-windows RNN state preservation on ST reconstruction when TL is employed during model inferencing. The TL achieves promising ST reconstruction and AD while substantially reducing the number of trainable parameters and providing better robustness against anomaly contamination in the training dataset. Our study demonstrates the efficacy of TL on ST data in overcoming the limitation of curated training data sparsity and computationally expensive model training.

4.3.1 Dataset Description

We utilized digi-occupancy data from the online DQM system to train and validate our models. Since the obtained data is from the LHC Run-2 collision experiment, we will describe the HCAL system and the data set configuration from 2018.

The digi-occupancy data sets were collected in 2018 during the LHC Run-2 collision experiment with the received luminosity per lumisection up to 0.4 pb^{-1}, and the number of events up to 2250. The source and target data sets contain three-dimensional digi-occupancy maps for the HE and HB subsystems of the HCAL, respectively (see Fig. 4.29). The relationships between the source and target data sets and tasks have been established to be important factors that impact the performance of TL [126]. The digi-occupancy map contains a particle hit count of the calorimeter readout channels for a given period of time. The HCAL covers a considerable volume of CMS and has a fine segmentation along three axes ($i\eta = [-32, \ldots, 32]$, $i\phi = [1, \ldots, 72]$ and $depth = [1, \ldots, 7]$), and a digi-occupancy measurement corresponds to a hit record of the readout channels located at the segmentation positions (see Fig. 4.29). The source system HE and the target system HB are subdetectors that cover different segments of the HCAL. One of the key differences between the HE and HB in the LHC collision experiment in 2018 was the frontend data acquisition optical-to-electrical technology—the HE employed Silicon photomultipliers (SiPMs) with QIE11 technology, and the HB utilized hybrid photodiode transducers (HPD) with QIE8. We summarize the comparison of the digi-occupancy maps of the source and target data sets in Table 4.6.



(a)

(b)

(c)

Figure 4.29: A sample digi-occupancy map (*year=2018, RunId=325170, LS=15)*: a) digi-occupancy map for the HBHE, b) the source system HE channels are placed in $|i\eta| = [16, 29]$, $i\phi = [1, 72]$, and $depth = [1, 7]$, and c) the target system HB channels are placed in $|i\eta| = [1, 16]$, $i\phi = [1, 72]$, and $depth = [1, 2]$. The missing sector at the top-left (HE) corresponds to the two failed RBXes in 2018.

4.3.2 Methodology

The GraphSTAD is a semi-supervised deep AE that detects an abnormality in the HCAL channels using reconstruction deviation scores on digi-occupancy maps from consecutive LSs. The AE combines CNN, GNN, and RNN to capture ST characteristics. Approximately 7K and 2.6K channels are in the digi-occupancy map of the HE and HB, respectively. We retrieved the readout channel to RBX mapping from the HCAL segmentation map in 2018 (see Fig. 1.13). We have explained the AD mechanism and model architecture of the GraphSTAD model in Section 4.2.2.

Table 4.6: Description of source and target data sets.

Dataset	Technology	Channels/RBX	No. of RBX	Segmentation	Sample Size
Source (HE)	SiPM	192	36	$\lvert i\eta\rvert = [16, 29]$, $i\phi = [1, 72]$, $depth = [1, 7]$	20K
Target (HB)	HPD	72	36	$\lvert i\eta\rvert = [1, 16]$, $i\phi = [1, 72]$, $depth = [1, 2]$	7K

Digi-occupancy Map Renormalization: We apply digi-occupancy renormalization in the data preprocessing stages to normalize the values for the variation in the luminosity and the number of event configurations of the collision experiments. The digi-occupancy (γ) map data of the HCAL varies with the received luminosity (β) and the number of events (ξ), and the per channel γ can range $\gamma = [0, \xi]$ (see Fig. 4.30). The β and γ are retrieved from different systems at CMS; directly accessing β for the real-time γ AD monitoring requires further effort. We renormalize the maps ($\gamma \rightarrow \hat{\gamma}$) by the ξ, usually equal to their maximum γ channel value, to achieve a consistent quantity interpretation of the γ maps. The remaining impact of β is left to be learned by the AD model.

The γ and ξ follow the β but not always. The non-linearity at the LHC—when ξ remains high while γ drops following β creates unpredictable spikes; renormalization of the γ with only the ξ does not entirely avoid the issue, and the spikes may affect the training performance temporal models. We employ additional reversible renormalization before and after invoking the AD model to mitigate the non-linearity of the digi-occupancy spatial data; the renormalization exploits the symmetric property of the $i\eta$ and *depth*—the γ values are less diverse along the $i\phi$ axis—and divides the input γ of the channels per each $i\eta$ and *depth* by their median values and reverse the action on the model output.

Anomaly Detection Mechanism: We denote the AE model of the AD system as $\mathcal{F}$. It takes ST data $\mathrm{X} \in \mathbb{R}^{T \times N_{i\eta} \times N_{i\phi} \times N_d \times N_f}$ as a sequence in a time window $t_x \in [t - T, t]$, where $N_{i\eta} \times N_{i\phi} \times N_d$ is the spatial dimension corresponding to the $i\eta$, $i\phi$, and *depth* axes, respectively, and N_f is the number of input variables; the $N_f = 1$ since we monitor only a digi-occupancy quantity in the spatial data. The $\mathcal{F}_{\theta,\omega} : \mathrm{X} \rightarrow \bar{\mathrm{X}}$—parametrized by θ and ω—attempts to reconstruct the input ST data X and outputs $\bar{\mathrm{X}}$. The encoder network of the model $E_\theta : \mathrm{X} \rightarrow \mathbf{z}$ provides low-dimension latent space $\mathbf{z} = E_\theta(\mathrm{X})$, and the decoder $D_\omega : \mathbf{z} \rightarrow \bar{\mathrm{X}}$ reconstructs the ST data from $\mathbf{z}$—$\bar{\mathrm{X}} = D_\omega(\mathbf{z})$ as:

$$\bar{\mathrm{X}} = \mathcal{F}_{\theta,\omega}(\mathrm{X}) = D_\omega(E_\theta(\mathrm{X})) \tag{4.31}$$

Anomalies can live short (impacting only a single digi-occupancy map) or persist over time (affecting a sequence of maps). Aggregated spatial reconstruction error e is calculated over a time window T using mean absolute error (MAE) to capture a

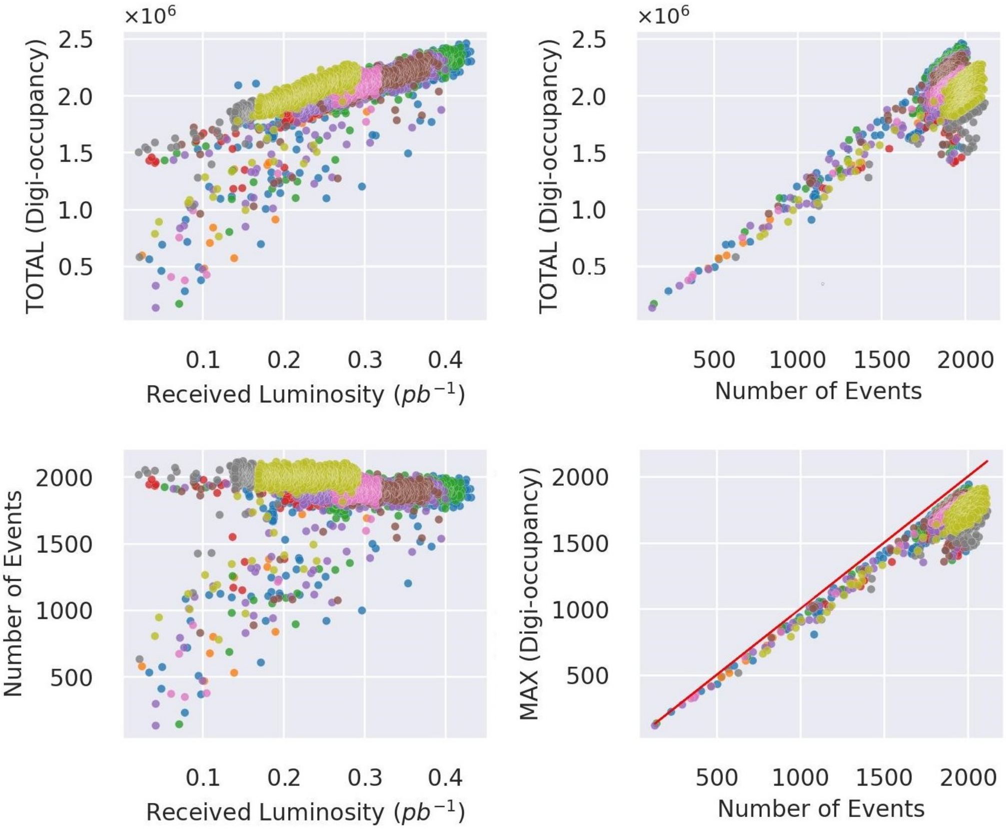


Figure 4.30: Digi-occupancy of the HB and run setting per LS—the received luminosity and the number of events. The different colors correspond to different runs.

time-persistent anomaly as:

$$
\begin{aligned}
e_i &= |x_i - \bar{x}_i| \\
e_{i,MAE} &= \frac{1}{T} \sum_{t'=t-T}^{t} e_i(t')
\end{aligned}
\tag{4.32}
$$

where $x_i \in \mathrm{X}$ and $\bar{x}_i \in \bar{\mathrm{X}}$ are the input and reconstructed digi-occupancy of the i^{th} channel. We standardize $e_{i,MAE}$ to homogenize the reconstruction accuracy variations among the channels when generating the anomaly score s_i as:

$$
s_i = \frac{e_{i,MAE}}{\sigma_i} \tag{4.33}
$$

where σ_i is the standard deviation of the $e_{i,MAE}$ on the training data set. The standardized anomaly score allows us to use a single AD decision threshold α for all the channels in the spatial map. The anomaly flags a_i are generated after applying α to the anomaly scores—$a_i = s_i > \alpha$. The α is a tunable parameter to control the detection sensitivity.

The use-case GraphSTAD AE model is made of CNN, GNN, and RNN networks; it employs CNN and GNN with a pooling mechanism to extract relevant features from spatial DQM data followed by RNN to capture temporal characteristics of the extracted features (see Fig. 4.18). The encoder integrates variational layer [83] at its end to regularize the AE by enforcing continuous and normally distributed latent representations [41, 97, 306, 352]. We normalize the spatial data per channel into a $[0, 1]$ for effective model training across the variations in calorimeter channels. We trained the AE on healthy digi-occupancy maps of the target HB system using an MSE loss function as:

$$\mathcal{L}_{MSE} = \frac{1}{M}\sum_{i}(x_i - \bar{x}_i)^2 \tag{4.34}$$

where x_i and $\bar{x}_i$ is the input and reconstructed $\hat{\gamma}$ of the i^{th} channel, respectively, and M is the total number of channels. The variational layer (denoted as VAE in Fig. 4.18) of the AE regularizes the training MSE loss using the KL divergence loss D_{KL} to achieve the normally distributed latent space as:

$$\mathcal{L} = \underset{W \in \mathbb{R}}{\operatorname{argmin}} \left\{ \mathcal{L}_{MSE} - \beta D_{KL} \left[\mathcal{N}(\mu_z, \sigma_z), \mathcal{N}(0, I) \right] + \rho \|W\|_2^2 \right\} \tag{4.35}$$

where $\mathcal{N}$ is a normal distribution with zero mean and unit variance, and $\|.\|$ is the *Frobenius norm* of L_2 regularization for the trainable model parameters W. The $\beta = 0.003$ and $\rho = 10^{-7}$ are tunable regularization hyperparameters. We employed *Adam* optimizer for training.

Transfer Learning Approach: Model parameter TL generally consists of four basic steps: 1) selection of a source task with a related modeling problem and an abundance of data where we can exploit the mapping knowledge from the inputs to outputs, 2) development of the source model that performs well in the source task, 3) transfer source model to target model where whole or part of the source model is employed as part of the target model, and 4) fine-tuning the target model on the target dataset if necessary. We present knowledge transfer on GraphSTAD AE models—trained on digi-occupancy maps of the source HE subsystem—to the target HB subsystem of the HCAL. Brute-forcing the knowledge from the source into the target irrespective of their divergence and thorough investigation of the several network building modules would cause certain performance degeneration—impacting the original data consistency in the target domain [191]. We have thus investigated several transferring cases when employing the TL on two principal model training phases—the initialization and training phases:

1. *Init mode*: the trainable network parameters (weights and bias) of the source model are transferred into the target model initialization. The target model is further trained on the target HB dataset—fine-tuning.

2. *Train mode*: The model parameters of the source model are directly reused as the final inference parameters of the target model; the parameters are frozen and excluded from fine-tuning on the target HB dataset.

Let $\mathcal{M}(\Psi, \Omega)$ be an AD model with parameters Ψ and Ω affected and not affected by TL, respectively, and $\mathcal{M}_e(\Psi_e, \Omega_e)$ and $\mathcal{M}_b(\Psi_b, \Omega_b)$ are the source and target models for the HE and HB, respectively. The TL modes $\mathcal{T}$ can be formulated mathematically as:

$$\begin{aligned} \underset{\text{init mode}}{\mathcal{T}} &: \mathcal{M}_b(\Psi_e, \Omega_b) \underset{\text{fine-tuning}}{\rightarrow} \mathcal{M}_b(\Psi_e^{'}, \Omega_b^{'}) \\ \underset{\text{train mode}}{\mathcal{T}} &: \mathcal{M}_b(\Psi_e, \Omega_b) \underset{\text{fine-tuning}}{\rightarrow} \mathcal{M}_b(\Psi_e, \Omega_b^{'}) \end{aligned} \tag{4.36}$$

where the superscript $'$ denotes the parameters that are updated after fine-tuning the $\mathcal{M}_b$ model on the target dataset.

We apply the above TL mechanisms to the different deep networks of the encoder and decoder of the GraphSTAD AE to study the impacts on ST digi-occupancy map reconstruction and AD accuracy (see Table 4.7). We analyze the effect of RNN state preservation within and across time windows. We further investigate training iteration epochs and learning rate scheduling methods. The discussion includes the impact of the TL on model accuracy, overfitting, and training stability.

Table 4.7: Transfer learning experiment configurations.

Config.	**Init Mode**		**Train Mode (on Target Dataset)**	
	Notation	**Description**	**Notation**	**Description**
1	RANDOM	Target model is initialized randomly	No-TL	Complete training (fine-tuning)
2	TL-4	Target model is initialized randomly, except the spatial learning networks (CNN and GNN) transferred from the source model	No-TL	
3			TL-1	The GNN of the encoder is frozen (not fine-tuned)
4			TL-2	The CNN of the encoder is frozen
5			TL-2$_d$	The CNN of the decoder is frozen
6			TL-3	The CNN and GNN of the encoder are frozen
7			TL-4	The CNN and GNN of the encoder, and the CNN of the decoder are frozen
8	TL-7	All the spatial and temporal learning networks (CNN, GNN, and RNN) of the target model are initialized by TL from the source model	TL-5	The CNN, GNN and the RNN of the encoder are frozen
9			TL-6	The CNN and GNN of the encoder, and the RNN of the decoder are frozen

TL - Transfer learning is applied.
TL-1: TL (ENCODER: [GNN]), TL-2: TL (ENCODER: [CNN]), TL-2$_d$: TL (DECODER: [CNN]), TL-3: TL (ENCODER: [CNN, GNN]), TL-4: TL (ENCODER: [CNN, GNN], DECODER: [CNN]), TL-5: TL (ENCODER: [CNN, GNN, RNN]), TL-6: TL (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]), TL-7: TL (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]), TL-8: TL (ENCODER: [CNN, GNN, RNN], DECODER: [CNN, RNN])

The implementation of parameter transferring on DL networks can be accomplished in two ways: 1) start with the source model and then reset—remove and add—the networks that are not included in the TL, and 2) start with the target model—random initialization—and update the parameter values of the networks included in the TL from the corresponding source networks. The first approach is commonly used for DL TL—employed for feature extraction; the approach may not be suitable to flexibly choose layers at different hierarchies, as the target models might have slight variations. Several configuration setups of the AE are derived from the spatial configuration of the input 3D map, which differs for the source HE and target HB systems—e.g., variation in the depth spatial dimension between HE and HB. We have found the second approach more convenient in our study, as we intend to apply TL on different networks of the encoder and decoder of the AE model.

4.3.3 Results and Discussion

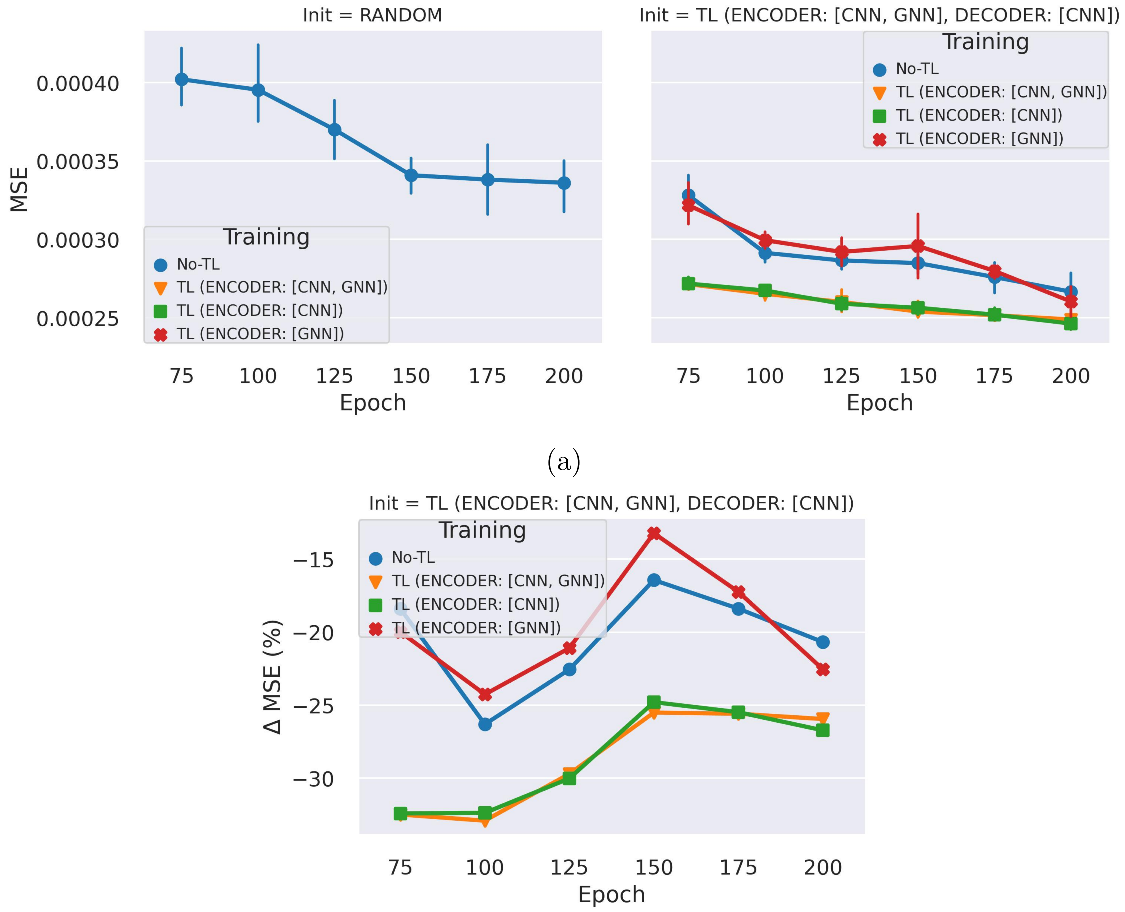
We will present the results of TL on reconstruction accuracy, trainable parameter reduction, and AD performance on the target HB digi-occupancy dataset. We applied TL for model initialization—*init mode*—without and with fine-tuning—*train mode*—on the target HB dataset. We trained the models on NVIDIA Tesla V100 with 4 GPUs using 4K digi-occupancy maps from LS 1 to 500 and evaluated the approximately 3K maps from LS from 500 to 1500. We utilized 20% of the training dataset (the last time stamps) for validation loss calculation during training to determine the best states for the models while training the models until the maximum epoch. We set the learning rate at 0.001 and the batch size at 6 to train the models with five lumisections per time window.

Spatio-Temporal Reconstruction Performance

We will discuss below the $\mathcal{L}_{MSE}$ performance of TL applied on spatial—CNNs and GNNs—and temporal—RNNs—learning networks. We will also briefly present a comparison of the learning rate scheduling mechanisms.

Transfer Learning on Spatial Learning Networks: We have assessed the transferability of DL model initialization and inference for the spatial learning networks—CNNs and GNNs—on both the encoder and decoder networks on different numbers of training epochs (see Fig. 4.31). The TL has reduced the reconstruction error $\mathcal{L}_{MSE}$ of healthy maps by 32.5% to 20.7% when the number of epochs is varied from 75 to 200 (see Figure 4.31b). The minimum gain of 13% is achieved at epoch 150, just before the performance of the no-TL model starts to saturate. The complete fine-tuning—TL for initialization followed by fine-tuning the whole network—accomplished around 20% improvement. The $\mathcal{L}_{MSE}$ generally decreases, while the relative TL gain roughly decreases as the epoch increases to 150. The results are not entirely unexpected; the DL models may tend to improve performance as the training epoch increases—reducing the gap caused by the difference in initialization and training mechanism. When the epoch increased beyond 150, the randomly initialized model (no-TL) achieves only slight improvement, whereas the $\mathcal{L}_{MSE}$ continues to drop for the TL models—increasing the relative gain of the TL. TL for initialization of all the spatial learning networks of the AE—*init mode* = TL-4—and fine-tuning only the decoder while freezing the encoder—*train mode*

= TL-3—achieves the best improvement—from 26% to 32.5%. The TL gain of the GNNs is limited compared to CNNs; the CNNs are the primary network that learns the input spatial data and has 15 times more parameters than GNNs in the use-case GraphSTAD AE model. Transferring and freezing the CNNs of the encoder—TL-2 and TL-3—exhibit stable performance on repeated experiments.



(a)

(b)

Figure 4.31: Reconstruction $\mathcal{L}_{MSE}$ evaluation of the TL on spatial networks across multiple epochs: a) Test MSE loss and the bars show the dispersion of five repeated experiments, and b) the average relative difference of the MSE loss with respect to the no-TL. The TL is applied using *init mode* = TL-4 and *train mode* = TL-1, TL-2, and TL-3. The no-TL model starts to saturate at *epoch* > 150.

Table 4.8 provides the average and best model ST reconstruction $\mathcal{L}_{MSE}$. Inference TL on the decoder networks without fine-tuning *train mode* = TL-2_d fails to reconstruct the target data adequately. In an AE architecture, the encoder maps the input into low dimensional latent space—information compression—while the decoder attempts to reconstruct—information expansion—the target data from the latent. The decoder networks thus require fine-tuning on the target dataset to adjust its parameters to the target reconstruction effectively. [190] investigates TL on DL for a univariate chaotic TS classification model; they argue that BN without fine-tuning limits the transferability of CNNs—the scaling and shifting parameters for

Table 4.8: Average ST reconstruction performance of TL at *epoch* = 200.

Init Mode	Train Mode	MSE Loss			ΔMSE w.r.t *Init*=RANDOM		
		Train	Val	Test	Train	Val	Test
Average Performance							
RANDOM	No-TL	2.650×10^{-04}	2.750×10^{-04}	3.361×10^{-04}	–	–	–
TL-4	No-TL	2.200×10^{-04}	2.250×10^{-04}	2.666×10^{-04}	-17.0%	-18.2%	-20.7%
TL-4	TL-3	1.775×10^{-04}	$\mathbf{1.775 \times 10^{-04}}$	2.489×10^{-04}	-33.0%	**-35.5%**	-25.9%
TL-4	TL-2	$\mathbf{1.725 \times 10^{-04}}$	1.800×10^{-04}	$\mathbf{2.463 \times 10^{-04}}$	**-34.9%**	-34.5%	**-26.7%**
TL-4	TL-1	2.075×10^{-04}	2.125×10^{-04}	2.604×10^{-04}	-21.7%	-22.7%	-22.5%
TL-4	TL-2_d	8.902×10^{-03}	7.380×10^{-03}	1.530×10^{-02}	3259.2%	2583.6%	4452.2%
Best Model (on Test Loss)							
RANDOM	No-TL	2.400×10^{-04}	2.600×10^{-04}	3.085×10^{-04}	–	–	–
TL-4	No-TL	2.100×10^{-04}	2.100×10^{-04}	2.569×10^{-04}	-12.5%	-19.2%	-16.7%
TL-4	TL-3	$\mathbf{1.700 \times 10^{-04}}$	$\mathbf{1.700 \times 10^{-04}}$	2.451×10^{-04}	**-29.2%**	**-34.6%**	-20.5%
TL-4	TL-2	$\mathbf{1.700 \times 10^{-04}}$	1.800×10^{-04}	$\mathbf{2.420 \times 10^{-04}}$	**-29.2%**	-30.8%	**-21.6%**
TL-4	TL-1	2.000×10^{-04}	2.100×10^{-04}	2.502×10^{-04}	-16.7%	-19.2%	-18.9%
TL-4	TL-2_d	8.88×10^{-03}	7.37×10^{-03}	1.5255×10^{-02}	3600.0%	2734.6%	4844.9%

TL-1: TL (ENCODER: [GNN]), TL-2: TL (ENCODER: [CNN]), TL-2_d: TL (DECODER: [CNN]), TL-3: TL (ENCODER: [CNN, GNN]), TL-4: TL (ENCODER: [CNN, GNN], DECODER: [CNN]), TL-5: TL (ENCODER: [CNN, GNN, RNN]), TL-6: TL (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]), TL-7: TL (ENCODER: [CNN, GNN, RNN], DECODER: [CNN, RNN])

BN and bias parameters are estimated from the training dataset and strongly correlate to the data. We have further studied TL on the decoder when the BN layer and the bias parameters of the CNNs are fine-tuned on the target dataset. The $\mathcal{L}_{MSE}$ is substantially reduced—accuracy improved—by 50% as compared to the frozen decoder (see Table 4.9). The reconstruction error is still 20 times higher than the without TL model—indicating the CNNs of the decoder also require fine-tuning to achieve reasonable accuracy. The results demonstrate the promising leverage of TL for an AE model initialization—on both feature extraction encoder and reconstruction decoder networks—whereas fine-tuning with the target data set is necessary for the decoder networks.

Table 4.9: Average ST reconstruction performance of TL—decoder with *init mode*=TL-4 at *epoch* = 200.

Train Mode	MSE Loss			ΔMSE w.r.t $TL-2_d$		
	Train	Val	Test	Train	Val	Test
TL-2_d	8.902×10^{-03}	7.380×10^{-03}	1.530×10^{-02}	–	–	–
TL-2_d / [BN]	$\mathbf{4.403 \times 10^{-03}}$	3.767×10^{-03}	$\mathbf{7.200 \times 10^{-03}}$	**-50.5%**	-48.9%	**-53.0%**
TL-2_d / [BN, BIAS]	4.405×10^{-03}	$\mathbf{3.760 \times 10^{-03}}$	7.354×10^{-03}	**-50.5%**	**-49.1%**	-51.9%

TL-2_d: TL (DECODER: [CNN]), TL-4: TL (ENCODER: [CNN, GNN], DECODER: [CNN])

Transfer Learning on Temporal Learning Networks: We have investigated TL on the temporal RNNs—LSTM layers—on both the encoder and decoder networks, together with the transferability of the spatial learning networks—CNNs and GNNs. We utilize the best performing TL from the above TL on spatial networks investigation—fine-tuning with *train mode* = TL-4 when conducting the TL experiment on temporal networks.

Table 4.10 presents the $\mathcal{L}_{MSE}$ when TL is applied to the ST networks at *epoch* =

Table 4.10: ST reconstruction performance of TL on temporal networks—without RNN states preservation—at *epoch* = 200.

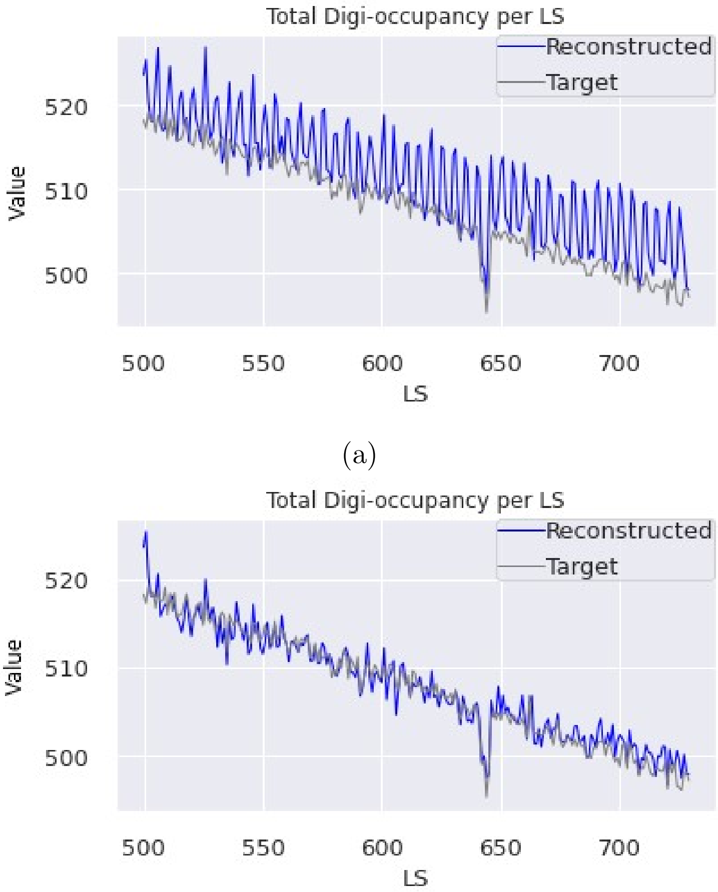
Init Mode	Train Mode	MSE Loss			ΔMSE w.r.t *Init*=RANDOM		
		Train	Val	Test	Train	Val	Test
Average Performance							
RANDOM	No-TL	2.650×10^{-04}	2.750×10^{-04}	3.361×10^{-04}	–	–	–
TL-4	TL-3	1.775×10^{-04}	$\mathbf{1.775 \times 10^{-04}}$	$\mathbf{2.489 \times 10^{-04}}$	-33.0%	**-35.5%**	**-25.9%**
TL-7	TL-5	1.775×10^{-04}	1.800×10^{-04}	2.496×10^{-04}	-33.0%	-34.5%	-25.7%
TL-7	TL-6	$\mathbf{1.725 \times 10^{-04}}$	4.325×10^{-04}	5.054×10^{-04}	**-34.9%**	57.3%	50.4%
Best Model (on Test Loss)							
RANDOM	No-TL	2.400×10^{-04}	2.600×10^{-04}	3.085×10^{-04}	–	–	–
TL-4	TL-3	1.700×10^{-04}	$\mathbf{1.700 \times 10^{-04}}$	$\mathbf{2.451 \times 10^{-04}}$	-29.2%	**-34.6%**	**-20.5%**
TL-7	TL-5	1.700×10^{-04}	1.800×10^{-04}	2.457×10^{-04}	-29.2%	-30.8%	-20.4%
TL-7	TL-6	$\mathbf{1.500 \times 10^{-04}}$	4.600×10^{-04}	4.697×10^{-04}	**-37.5%**	76.9%	52.2%

TL-3: TL (ENCODER: [CNN, GNN]), TL-4: TL (ENCODER: [CNN, GNN], DECODER: [CNN]), TL-5: TL (ENCODER: [CNN, GNN, RNN]), TL-6: TL (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]), TL-7: TL (ENCODER: [CNN, GNN, RNN], DECODER: [CNN, RNN])

200. We have evaluated models with RNN states preserved only within five maps in a time window—states are reset for each time window. Transferring RNNs has not accomplished performance gain. When the TL involves freezing the RNNs of the decoder for inference on the target data—*train mode*: TL-6—the performance suffered substantially—increasing the test $\mathcal{L}_{MSE}$ by more than 50% despite the reduction of the training $\mathcal{L}_{MSE}$ by 34%. A close investigation has revealed that the issue lies with RNN state resetting during inference. Fig. 4.32 illustrates that the model is struggling to reconstruct the first time step in the time windows (TWs). The reason is that the model relies on only the first input map for the first time step with reset states, while the states are adjusted and improved for the following maps (see Fig. 4.32a). This behavior is not entirely unexpected; the RNNs are trained on the source dataset, and employing them directly on the target data set—without taking advantage of the localized temporal information—for the first map in a TW—would be challenging. We have further evaluated the models by preserving the RNN states across time windows that leverage the model reconstruction accuracy; it utilizes previous states even for the first maps in the TWs (see Fig. 4.32b).

Fig. 4.33 illustrates the $\mathcal{L}_{MSE}$ on multiple epochs for the RNN networks with and without RNN states preservation across TWs. The plots illustrate a significant improvement by preserving the states on the frozen decoder RNNs—*train mode* = TL-6; higher gaps are also observed for lower epochs among repeated experiments (five times) but the stabilization gets better at higher epochs. State preservation across TW has a limited impact when the target dataset fine-tunes the decoder RNNs—the *train mode*: TL-5. We have summarized the best performance in Table 4.11.

Applying Learning Rate Scheduling: The models reach saturation after *epoch* > 150 as illustrated in the previous plots in Fig. 4.31a and Fig. 4.33b for the models trained without TL and with TL, respectively. Learning rate (LR) scheduling mechanisms—e.g., lowering the LR when the loss gets flat or fast con-



(a)

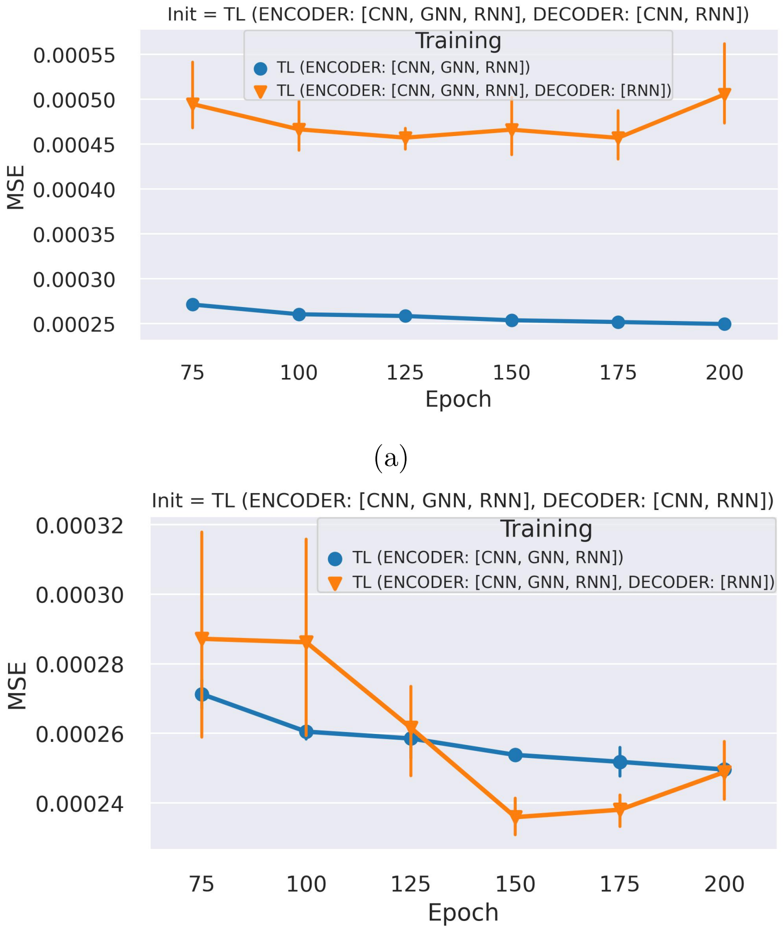
(b)

Figure 4.32: Digi-occupancy map reconstruction of the model trained with TL-6: a) without, and b) with LSTM states preservation across time-windows. The AE operates on ST γ maps, but the curves in these plots correspond to the aggregate renormalized γ per LS to illustrate the model's reconstruction performance in handling the fluctuation across lumisections.

vergence methods—could improve the performance to mitigate training saturation. We have investigated the impact of scheduling on the TL by training the model with super-convergence *one-cyclic* LR scheduling [356] at *epoch* = 200. The LR scheduling sets the LR according to a one-cycle policy that anneals the LR from an initial LR to some maximum LR ($max\_lr = 0.001$) and then from that maximum LR ($min\_lr = 4 \times 10^{-7}$) to some minimum LR. We utilized a cosine annealing mechanism along with the other settings of the scheduler—provided in Table 4.12—for our experiment. We kept the default values of the remaining hyperparameters of the scheduler[10].

Table 4.13 shows that the LR scheduling has improved the $\mathcal{L}_{MSE}$ compared to the fixed LR (provided in Table 4.11) by 19% and 13% for *init mode* = RANDOM, and the *train mode* = TL-6 models, respectively. With respect to the model without TL, the relative improvement from TL is approximately 9% with the LR scheduling, which is lower than the 22.6% achieved with the fixed LR. The results are in accord

[10] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html



(a)

(b)

Figure 4.33: Reconstruction $\mathcal{L}_{MSE}$ evaluation of the TL on the RNN (test-set). The TL is applied to initialize the encoder's and decoder's CNN, GNN, and RNN networks and trained with TL on the RNNs. Test MSE loss on a) non-preserved LSTM states that reset for each time window, and b) preserved LSTM states across consecutive time windows. The bars show the dispersion of five repeated experiments.

with Fig. 4.31b—following the projection of closing in the performance difference as the number of epochs increases past *epoch* > 150 with resolved saturation on the *init mode* $=$ RANDOM. The cyclic LR scheduling method requires more configuration tuning effort to improve the performance depending on the model and dataset compared to fixed LR or other simpler LR scheduling approaches.

Anomaly Detection Performance

Machine learning studies performed thus far in the DQM system of CMS experiment primarily employed the simulated anomalies data to evaluate the effectiveness of the developed AD models [39]; a small fraction of the DQM data is affected by real anomalies and is limited to be used for model validation. We have validated the AD models on synthetic anomalies simulating real channel anomalies of the HCAL. We generated synthetic anomalies simulating *dead*, *hot*, and *degraded* channels, and

Table 4.11: ST reconstruction performance of TL with RNN state preservation.

Init Mode	**Training Mode**	**MSE Loss**		**Trainable Parameters**
		Value	Δ(%) **w.r.t** ***Init***=RANDOM	**Reduction (%)**
At *epoch* = 75				
RANDOM	No-TL	3.826×10^{-04}	–	
TL-4	No-TL	3.180×10^{-04}	-16.9%	0.00%
TL-4	TL-2	2.686×10^{-04}	-29.8%	-2.23%
TL-4	TL-1	3.082×10^{-04}	-19.5%	-0.17%
TL-4	TL-3	2.705×10^{-04}	-29.3%	-2.39%
TL-7	TL-5	2.667×10^{-04}	-30.3%	-8.38%
TL-7	TL-6	$\mathbf{2.577 \times 10^{-04}}$	**-32.6%**	**-97.77%**
At *epoch* = 200				
RANDOM	No-TL	3.085×10^{-04}	–	–
TL-4	No-TL	2.569×10^{-04}	-16.7%	0.00%
TL-4	TL-2	2.420×10^{-04}	-21.6%	-2.23%
TL-4	TL-1	2.502×10^{-04}	-18.9%	-0.17%
TL-4	TL-3	2.451×10^{-04}	-20.5%	-2.39%
TL-7	TL-5	2.457×10^{-04}	-20.4%	-8.38%
TL-7	TL-6	$\mathbf{2.389 \times 10^{-04}}$	**-22.6%**	**-97.77%**

TL-1: TL (ENCODER: [GNN]), TL-2: TL (ENCODER: [CNN]), TL-3: TL (ENCODER: [CNN, GNN]), TL-4: TL (ENCODER: [CNN, GNN], DECODER: [CNN]), TL-5: TL (ENCODER: [CNN, GNN, RNN]), TL-6: TL (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]), TL-7: TL (ENCODER: [CNN, GNN, RNN], DECODER: [CNN, RNN])

Table 4.12: Hyperparameter setting of one-cyclic LR scheduler.

Hyperparameter	**Value**	**Description**
max_lr	0.001	Upper learning rate boundaries in the cycle.
steps_per_epoch	*train_data_size/batch_size*	The number of steps per epoch to train for.
total_steps	*steps_per_epoch* × *epochs*	The total number of steps in the cycle.
div_factor	25	Determines the initial learning rate via $initial\_lr = max\_lr/div\_factor$.
final_div_factor	10^2	Determines the minimum learning rate via $min\_lr = initial\_lr/final\_div\_factor$.

injected them into healthy digi-occupancy maps of the test dataset. We formulate the simulated channel anomalies using:

$$\gamma_a = R_D \gamma_h, \;\; R_D \neq 1 \tag{4.37}$$

where γ_a and γ_h are the digi-occupancy of the generated anomaly channel and its corresponding expected healthy reading, respectively. The R_D is the degradation factor, and the simulated anomalies are defined as:

$$\begin{aligned} Dead\ Channel &: \gamma_a = R_D\gamma_h = 0, \;\; R_D = 0 \\ Degraded\ Channel &: 0 < \gamma_a = R_D\gamma_h < \gamma_h, \;\; 0 < R_D < 1 \\ Noisy - Hot\ Channel &: \gamma_h < \gamma_a = R_D\gamma_h \leq \xi, \;\; R_D > 1 \\ Fully - Hot\ Channel &: \gamma_h < \gamma_a = \xi \end{aligned} \tag{4.38}$$

The algorithm that generates the synthetic anomaly samples involves three steps: 1) selection of a random set of LSs from the test set, 2) random selection of spatial locations φ for each LS, where $\varphi \in [i\eta \times i\phi \times depth]$ on the HB axes (see Fig.

Table 4.13: ST reconstruction performance of TL with learning rate scheduling mechanism at *epoch* = 200.

Init Mode	**Training Mode**	**MSE Loss**	
		Value	**Δ(%) w.r.t *Init*=RANDOM**
RANDOM	No-TL	2.500×10^{-04}	–
TL-4	No-TL	2.400×10^{-04}	-4.0%
TL-4	TL-3	2.460×10^{-04}	-1.6%
TL-7	TL-5	$\mathbf{2.283 \times 10^{-04}}$	**-8.7%**
TL-7	TL-6	$\mathbf{2.286 \times 10^{-04}}$	**-8.6%**

TL-3: TL (ENCODER: [CNN, GNN]), TL-4: TL (ENCODER: [CNN, GNN], DECODER: [CNN]), TL-5: TL (ENCODER: [CNN, GNN, RNN]), TL-6: TL (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]), TL-7: TL (ENCODER: [CNN, GNN, RNN], DECODER: [CNN, RNN])

4.29c), and 3) injection of the simulated anomalies into digi-occupancy maps of the LSs. The simulated anomalies include *dead*, *degraded*, *noisy-hot*, and *fully-hot* channels. We kept the same spatial locations of the generated different anomaly types for consistency. We evaluated the performance on several classification metrics using three anomaly thresholds set to capture 90%, 95%, and 99% of the injected anomalies.

The AD evaluation covers 14K digi-occupancy maps—2K maps for each anomaly type—for the dead ($R_D = 0\%$), degrading channel anomalies ($R_D = [80\%, 60\%, 40\%, 20\%]$), noisy-hot ($R_D = 200\%$), and fully-hot ($\gamma_a = \xi$) channels. We focus on persisting channel anomalies that affect consecutive maps in a time window of LSs. We thus processed 70K digi-occupancy maps—including five history maps in the time window for each of the 14K maps; we generated 1.17% abnormal channels in the 70K maps.

We compare the AD performance of models without TL and the best with TL given in Table 4.13—*without-TL*: *init mode* = RANDOM, and *with-TL*: *train mode* = TL-6. Table 4.14 presents the AD accuracy of the models on the dead, degraded, fully-hot, and noisy-hot channel abnormalities. Both models perform well in the *area under the receiver operating characteristic curve* (AUC) and *false positive rate* (FPR). The FPR exhibits slight variance between the two models; the TL model significantly improves dead and fully-hot channel detection but performs slightly lower for noisy-hot channels. Fig. 4.34 illustrates the FPR score across all the anomaly types where the TL model outperforms the without the TL model in $R_D < 1$. The relatively lower precision at 70% for the $R_D = 80\%$ demonstrates that there are still a few anomalies challenging to catch despite the FPR being very low due to the accurate classification of numerous healthy channels (see Fig. 4.35). A channel operating at 80% is mostly an inlier—overlaps with the healthy operating ranges—and detecting such anomaly becomes even more difficult when the expected γ of the channel is very low. The FPR has significantly improved by 80% when the amount of the captured anomaly is reduced to 95%. Fig. 4.36 and Fig.4.37 demonstrate the ability of our algorithm to locate the anomalies on a sample that has been injected with different anomaly types in some channels ($4 < i\eta < 11$ and $11 < i\phi < 19$); the TL model accomplishes better localization on the fully-hot channels with less

Table 4.14: AD performance of TL on time-persistent abnormal channels.

Anomaly Type	**FPR (90%)**	**FPR (95%)**	**FPR (99%)**	**AUC**
Without-TL Model: (*Init*=RANDOM, *Training*=No-TL)				
Degraded Channel ($R_D = 80\%, \gamma_a = R_D\gamma_h$)	6.281×10^{-04}	1.519×10^{-03}	8.741×10^{-03}	0.993
Degraded Channel ($R_D = 60\%, \gamma_a = R_D\gamma_h$)	5.991×10^{-05}	1.438×10^{-04}	8.242×10^{-04}	**1.000**
Degraded Channel ($R_D = 40\%, \gamma_a = R_D\gamma_h$)	4.881×10^{-05}	5.628×10^{-05}	1.466×10^{-04}	**1.000**
Degraded Channel ($R_D = 20\%, \gamma_a = R_D\gamma_h$)	5.870×10^{-05}	6.273×10^{-05}	7.342×10^{-05}	**1.000**
Dead Channel ($R_D = 0\%, \gamma_a = 0.0$)	6.636×10^{-05}	7.080×10^{-05}	8.331×10^{-05}	**1.000**
Fully-Hot Channel ($\gamma_a = \xi, \gamma_a \neq \gamma_h$)	1.220×10^{-04}	1.317×10^{-04}	1.606×10^{-04}	**1.000**
Noisy-Hot Channel ($R_D = 200\%, \gamma_a = R_D\gamma_h$)	**3.300×10^{-04}**	**5.732×10^{-04}**	**1.765×10^{-03}**	**1.000**
With-TL Model: (*Init*=TL-6, *Training*=TL-6)				
Degraded Channel ($R_D = 80\%, \gamma_a = R_D\gamma_h$)	**5.019×10^{-04}**	**1.320×10^{-03}**	**6.527×10^{-03}**	**0.996**
Degraded Channel ($R_D = 60\%, \gamma_a = R_D\gamma_h$)	**2.118×10^{-05}**	**9.642×10^{-05}**	**8.141×10^{-04}**	**1.000**
Degraded Channel ($R_D = 40\%, \gamma_a = R_D\gamma_h$)	**1.614×10^{-06}**	**4.034×10^{-06}**	**7.161×10^{-05}**	**1.000**
Degraded Channel ($R_D = 20\%, \gamma_a = R_D\gamma_h$)	**1.614×10^{-06}**	**3.833×10^{-06}**	**8.472×10^{-06}**	**1.000**
Dead Channel ($R_D = 0\%, \gamma_a = 0.0$)	**1.815×10^{-06}**	**4.236×10^{-06}**	**8.472×10^{-06}**	**1.000**
Fully-Hot Channel ($\gamma_a = \xi, \gamma_a \neq \gamma_h$)	**0.000**	**6.051×10^{-07}**	**3.631×10^{-05}**	**1.000**
Noisy-Hot Channel ($R_D = 200\%, \gamma_a = R_D\gamma_h$)	1.380×10^{-03}	2.143×10^{-03}	4.099×10^{-03}	**1.000**

dispersion in its anomaly score values.

The distribution of the reconstruction error—shown in Fig. 4.38—provides further illustration of the overlap region between the healthy and faulty channels at the different degradation rates. We have observed a slight increase in the reconstruction error of the healthy channels as the anomaly strength increases for the abnormal channels—the R_D is farther away from 100%; it is more pronounced when introducing hot channel anomalies with $R_D = 200\%$. Close investigation reveals that the healthy channels have higher anomaly scores—filter out from the anomaly injection $\gamma_a = R_D\gamma_h > \xi$—due to their proximity to the abnormal channels (see Fig. 4.39 and Fig. 4.40). Since channels belonging to the same RBX are positioned in proximity in the HB segmentation and share characteristics, the GraphSTAD AE exploits the correlation for spatial data reconstruction. The exacerbating effect in the presence of the hot channels indicates the model is giving prominence to the channel with higher values to perform the map reconstruction.

The performance on the dead channels is another major difference between the *without-TL* and *with-TL* models. The overlap tails decrease on reconstruction error distributions of the normal and abnormal channels—the degraded channels as the R_D decreases—in both models (see Fig. 4.38). The AD performance slightly deteriorates for the dead channels ($R_D = 0\%$) compared to the degraded channel at $R_D = 20\%$ defying expectation (see Table 4.14). Fig. 4.38 illustrates the reconstruction error of most anomalies increases—enabling enhanced separation between the normal and anomaly channels except for a few dead channels that increase the FPR. The reconstruction error of the *without-TL* model drops to zero for the dead channels, although the channels have higher reconstruction errors at $R_D = 20\%$; this is caused by the presence of real dead channels in the training dataset at the location of $[i\eta \in [-16, -15, -13],\ i\phi = 8,\ depth = 1]$ (see Fig. 4.41). Fig. 4.41 depicts that the *without-TL* model reconstructs the real dead channels as normal with low reconstruction error, whereas the *with-TL* provides a high error which sig-

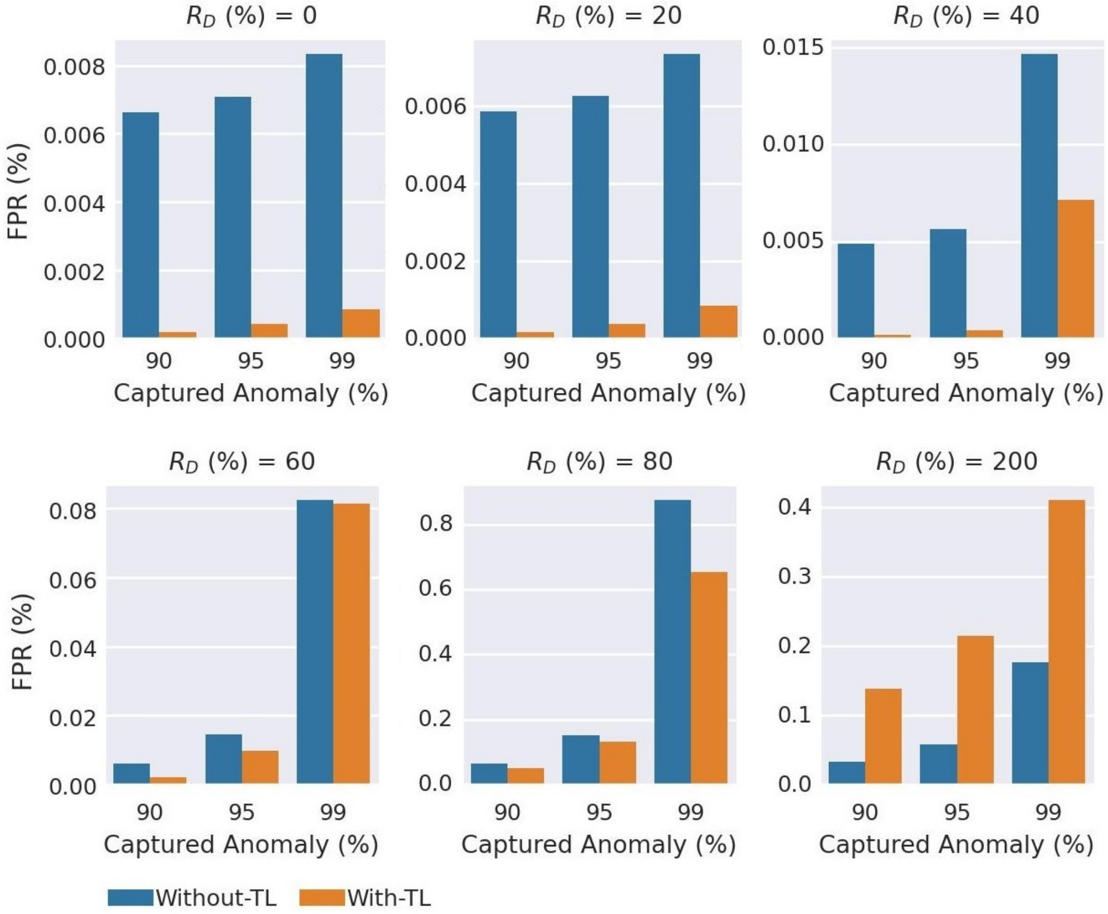


Figure 4.34: AD FPR on *dead*, *degraded* and *noisy-hot* channel. The anomaly flags are generated using thresholds that capture 90%, 95%, and 99% of the anomaly channels.

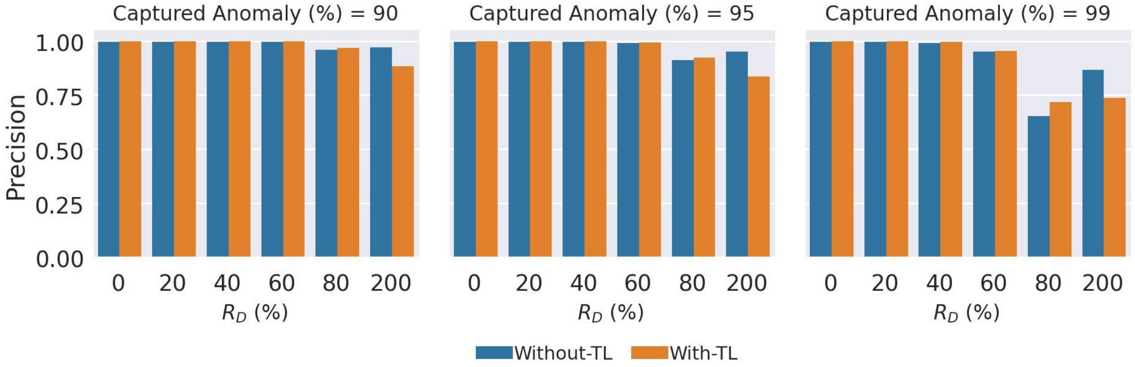


Figure 4.35: AD *Precision* on *dead*, *degraded* and *noisy-hot* channels. The anomaly flags are generated using thresholds that capture 90%, 95%, and 99% of the anomaly channels.

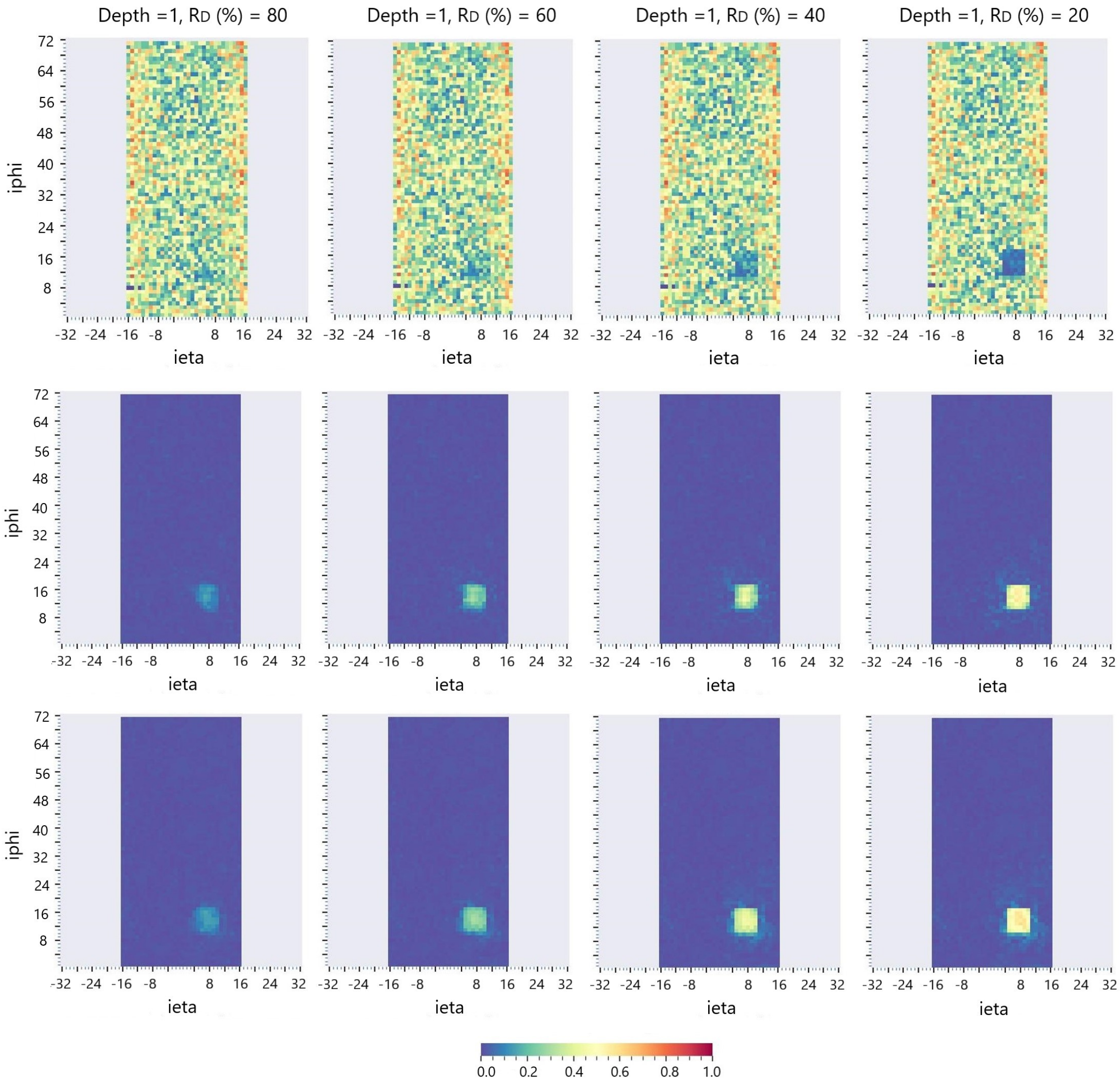


Figure 4.36: Spatial reconstruction error ($e_{i,MAE}$) maps on a sample digi-occupancy map at *depth* $= 1$ with degraded anomalies. Plots (top to bottom): digi-occupancy map with simulated channel anomalies, the reconstruction error maps of the *without-TL* model, and *with-TL* model. The anomaly region is localized well with proportional strength to the severity of the anomaly in both models.

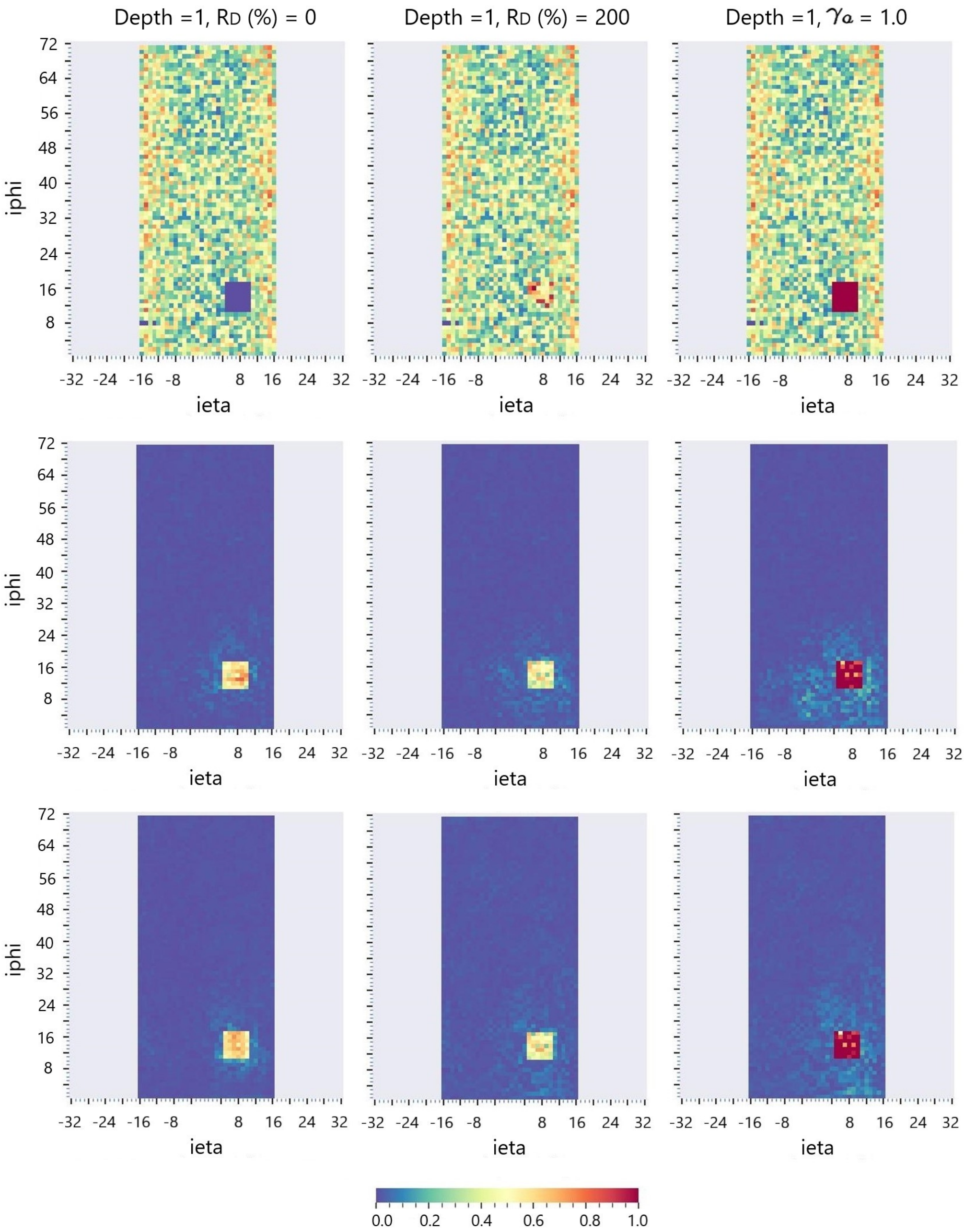
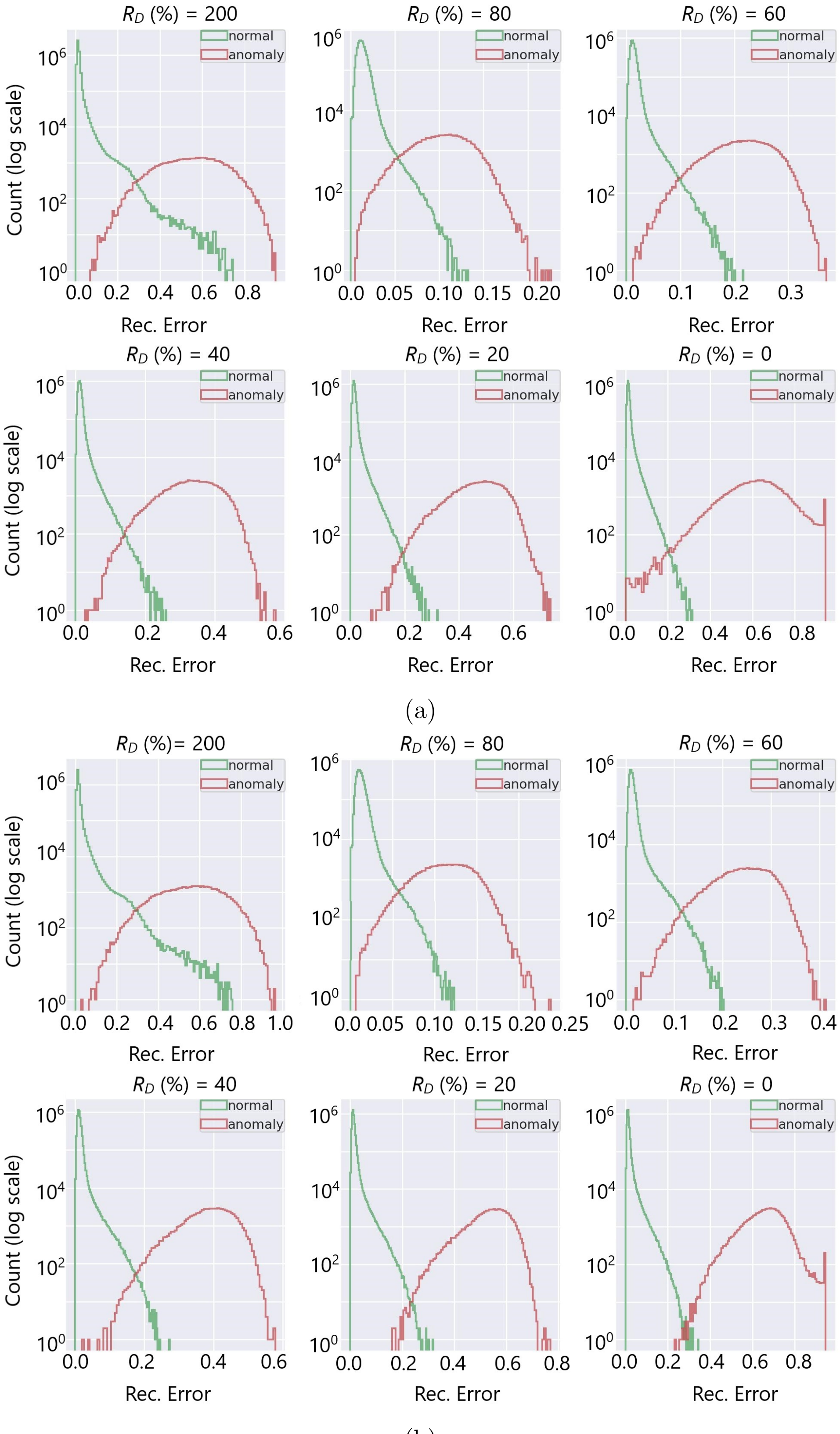


Figure 4.37: Spatial reconstruction error ($e_{i,MAE}$) maps on a sample digi-occupancy map at *depth* = 1 with dead, noisy-hot, and fully-hot anomalies. Plots (top to bottom): digi-occupancy map with simulated channel anomalies, the reconstruction error maps of the *without-TL* model, and *with-TL* model.



(a)

(b)

Figure 4.38: Reconstruction error ($e_{i,MAE}$) distribution of healthy and anomalous channels at different channel degradation rates—excluding the real anomalies. The models are a) *without-TL*, and b) *with-TL*. The overlap region decreases substantially as the channel deterioration increases for $R_D < 100\%$.

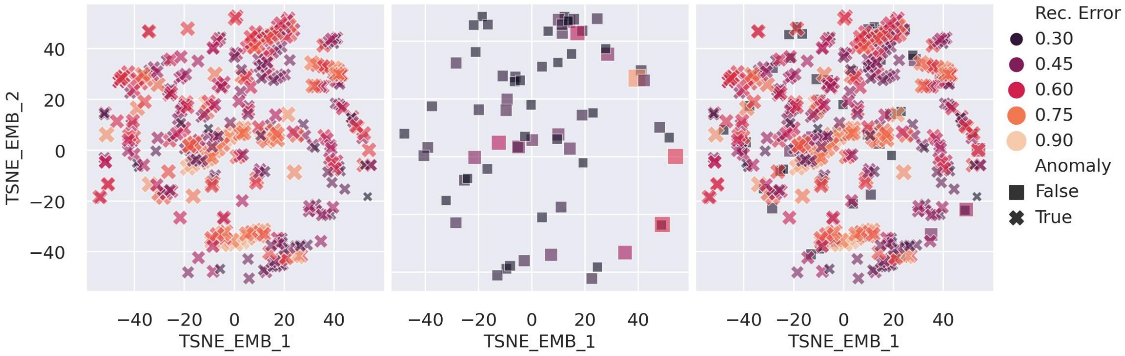
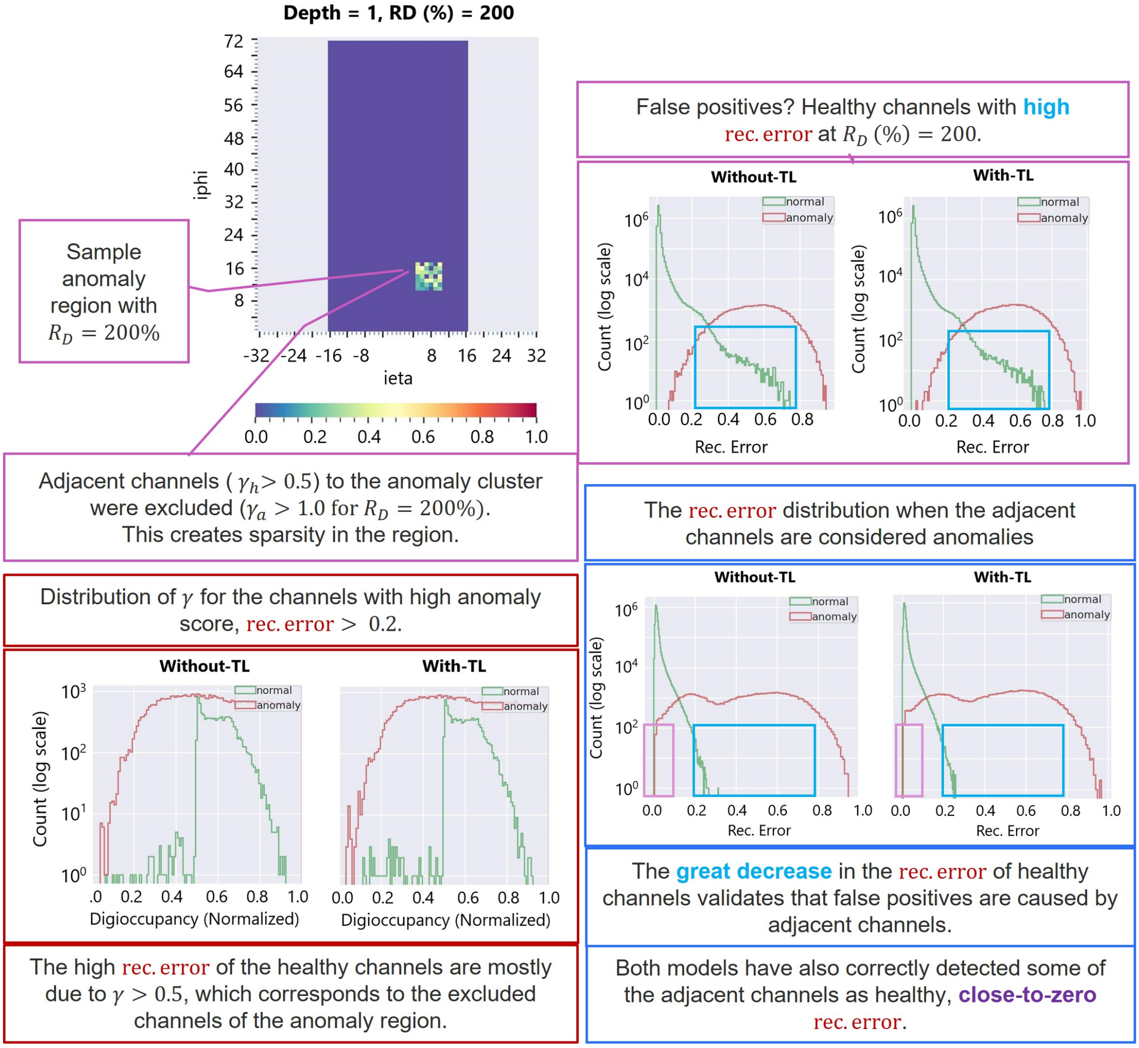


Figure 4.39: 2D Location embedding of high reconstruction error channels—from the *with-TL* model—in the presence of noisy-hot channel anomalies. We applied t-SNE [16] on the spatio-temporal locations (LS, $i\eta$, $i\phi$, and *depth*) to generate the 2D representation. Healthy channels with high reconstruction error scores are located near the anomalous channels.



Sample anomaly region with $R_D = 200\%$

False positives? Healthy channels with **high** rec. error at $R_D\,(\%) = 200$.

Adjacent channels (γ_h> 0.5) to the anomaly cluster were excluded ($\gamma_a > 1.0$ for $R_D = 200\%$). This creates sparsity in the region.

The rec. error distribution when the adjacent channels are considered anomalies

Distribution of γ for the channels with high anomaly score, rec. error > 0.2.

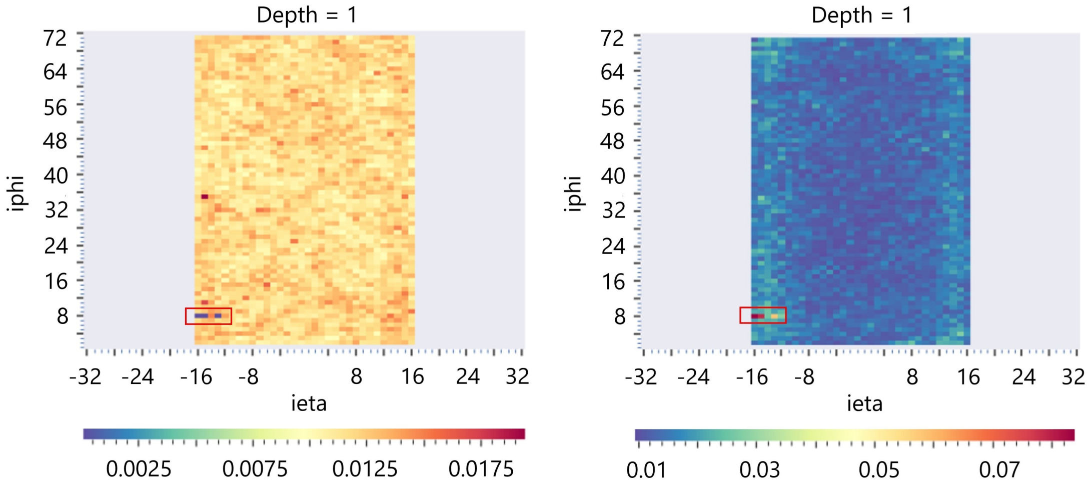
The **great decrease** in the rec. error of healthy channels validates that false positives are caused by adjacent channels.

The high rec. error of the healthy channels are mostly due to $\gamma > 0.5$, which corresponds to the excluded channels of the anomaly region.

Both models have also correctly detected some of the adjacent channels as healthy, **close-to-zero** rec. error.

Figure 4.40: Proximity effect explanation on false positive on *noisy-hot* channel anomaly ($R_D = 200\%$) detection. The healthy channels with higher anomaly scores are the ones filtered out from the anomaly injection $\gamma_a = R_D\gamma_h > \xi$ and generate a high score due to their proximity to the abnormal channels.

nifies it is detecting the channels as anomalies. The *without-TL* model reconstructs the simulated dead anomalies as healthy and fails to detect the anomalies when the simulated anomalies are injected into locations near the real abnormal channels. The *with-TL* has thus achieved substantially better detection on the dead channels (see Fig. 4.38b). The results demonstrate transfer learning robustness when a semi-supervised model's training dataset is contaminated with real anomalies.



(a)

(b)

Figure 4.41: Average reconstruction error map of the training dataset at $depth = 1$ for a) the *without-TL* model, and b) the *with-TL* model. The real dead HB channels (at $[i\eta = [-16, -15, -13], i\phi = 8, depth = 1]$) are highlighted in the red boxes. *without-TL* model reconstructs the real dead channels as normal with low error, whereas *with-TL* produces a high error detecting the channels as anomalies.

4.3.4 Summary

We have studied transfer learning on a spatio-temporal semi-supervised anomaly detection model. We have successfully transferred the spatio-temporal AD model—employing convolutional, graph, and recurrent neural networks in an autoencoder architecture—from the source HCAL Endcap to the target HCAL Barrel subdetector for data quality monitoring. The study has provided insights into several transfer learning scenarios at the initialization and training phases. We have successfully applied TL to the encoder feature extraction networks and inner networks of the decoder. The TL has achieved a promising ST reconstruction and AD performance while providing a substantial reduction in trainable parameters. We have also demonstrated that the TL can have better robustness against contamination in the training dataset. The study indicates that TL can facilitate machine learning development with limited clean training data and when expensive model training on large data sets is costly or time-consuming. The choice of model training settings, such as the number of iterations, learning rate schedule, and recurrent neural network state preservation during inference, can impact the TL performance besides the similarity between the source and target data sets.

Chapter 5

Anomaly Prediction

This chapter presents our study on deep learning modeling for multivariate anomaly prediction for infrastructure monitoring.

5.1 Long Horizon Anomaly Prediction in Multivariate Time Series with Causal Autoencoders

Complex industrial systems employ predictive maintenance to foresee anomalies before major system faults or ultimate breakdown. The existing efforts on industry 4.0—the fourth industrial revolution—predictive monitoring focus on semi-supervised anomaly detection with limited robustness for large systems that are often accompanied by uncleaned and unlabeled data. We address the challenge of predicting anomalies through data-driven end-to-end deep learning models using early warning symptoms on multivariate time series sensor data. We introduce a long multi-timestep anomaly prediction system (AnoP) based on unsupervised attention-based causal residual networks to raise alerts for anomaly prevention. The experimental evaluation on large data sets from detector health monitoring of the Hadron Calorimeter validates the promising efficacy of the proposed approach. The AnoP predicts around 60% of the anomalies up to seven days ahead, and most missed anomalies are abnormalities with unpredictable noisy-like behavior. The proposed approach has discovered previously unknown anomalies in the calorimeter's sensors.

Predicting faults in the HCAL detector electronics is essential for maintaining the quality of physics data acquisition. Machine learning algorithms have been explored through AD for system monitoring automation of the LHC accelerator and detector systems [39, 41, 52, 56, 69]. Forecasted anomalous behavior in a large set of detector sensors can indicate future performance issues—escaping the DCS and DSS monitoring. For example, the gradual decrease in the monitored received signal strength indicator (RSSI) current—proportional to the received light at the front end from the back end optical communication links—preceded control communication loss during operation in 2018 and 2019 [68, 357]. The RSSI was not actively monitored, and trend drifts—depicted in Fig. 5.1—could have been predicted. Our proposed approach in this study attempts to detect such anomalies from early signs before they affect data quality or result in data loss.

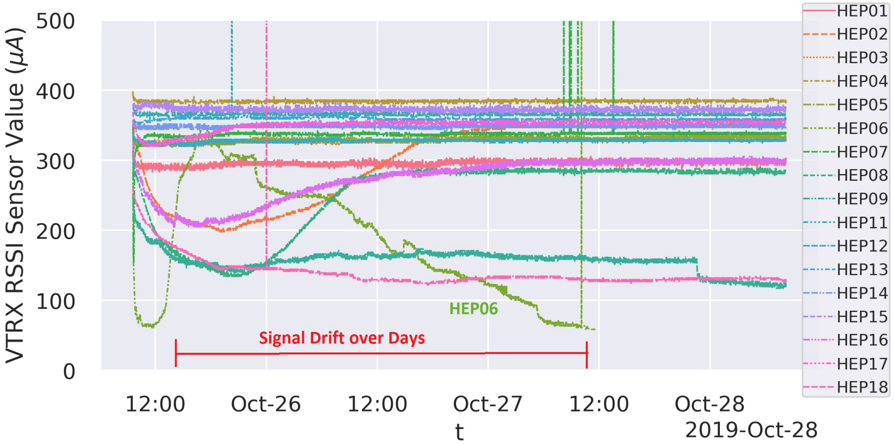


Figure 5.1: Gradual drifting anomalies on RSSI before ngCCM lost communication in 2019. A strong decay over three days is illustrated for the HEP06 RBX.

Modern industrial systems utilize sensors to monitor physical quantities, such as voltages, currents, flows, temperature, pressure, etc. These measurements monitor the system state by detecting deviations from normal operating conditions. Predictive maintenance (PdM) approaches—as one of the pillars of industry 4.0—are expected to improve asset availability by actuating early maintenance before major system faults [156]. Anomaly prediction (AP)—a subtask of PdM—extends AD and focuses on predicting anomalies from early symptoms. It has cost-saving potential for large complex systems through prevention of unforeseen system faults, unplanned downtimes, and maintenance [143,155–157,162]. Most of the data-driven PdM models in the literature employ supervised approaches that require prior labeled anomalies and are limited to short-range predictions [143,156–159,162].

We strive to predict anomalies through data-driven machine learning models from early warning patterns on unlabeled multivariate time series data sets. Capturing anomalies that persist for substantial periods—often manifested in decaying or growing trends, strange dips, or peaks—is the primary focus of our study. We propose AnoP—an end-to-end anomaly prediction system—using an integration of unsupervised long sequence time series forecasting and anomaly detection mechanisms. The proposed system consists of a pipeline of MTS autoencoder models—a long horizon sequence-to-sequence (S2S) time series forecasting (TSF) model and a TS AD model. The underlying concept employs a TSF model—trained on the interaction of multivariate sensor signals—to predict future temporal segments and then utilizes the AD model to evaluate the predicted signals for potential anomalies. Since additive outliers—transient and interpreted on short time scales—are generally unpredictable, our study aims at forecasting anomalous temporary changes that persevere for a certain period—multiple time steps [126].

We validate the AnoP system to predict anomalies from the multivariate diagnostics sensor data and leverage the health monitoring prognostics of the HE subdetector. We evaluate AnoP in predicting temporal discords using various long sequence horizons on thirty-four readout boxes. We incorporate an evaluation of the forecasting accuracy of the TSF model. The results demonstrate that the proposed system reveals new anomalies—never captured before in the HCAL. The key contributions of our study are highlighted below:

1. We present a data-driven unsupervised anomaly prediction mechanism from a heterogeneous multivariate time series sensor dataset.

2. We introduce a time block-based S2S TSF model that captures temporal causal interactions for long sequence multivariate time series prediction.

3. We discuss a first study on early prognostics through data-driven methods for HCAL-RBX monitoring from diagnostic sensor data.

5.1.1 Dataset Description

We utilized front-end electronics sensor data from the HCAL—recorded for detector health monitoring and diagnostic purposes. The data is from the ngCCM of the HE subdetector and was collected in 2018 using the ngCCM server. The ngCCM server—a software that handles access to the ngCCM—enables communication between detector monitoring services and the front-end electronics. The data set contains approximately one-minute 86M samples—from September to December 2018 from 34 active RBXes—i.e., HEP01—18 and HEM01—18, excluding HEM15 and HEM16. The data consists of 28 sensors from current, voltage, and optical power measurements of various components of the ngCCM. We downsampled the data into hourly intervals by averaging to capture the relevant temporal information in a broader time interval.

5.1.2 Methodology

Our proposed AnoP system is composed of two MTS models combined in a pipeline: 1) a multi-timestep TSF model, and 2) an AD model (see Fig. 5.2). We will mainly discuss below the mathematical formulation and architectures for the TSF model of the AnoP; we have already explained the AD model in the previous Section 4.1.2. we will also elaborate on the data preprocessing, preparation of training data sets, and model training.

Multivariate Multi-timestep Forecasting Model: We present a robust attention-based S2S dynamic conditional decoding mechanism for long sequence forecasting. A forecasting model must cope with two challenges for anomaly prediction tasks: 1) it should predict the deviating signals belonging to anomalies from their early fluctuation patterns, and 2) it should also quickly adjust its prediction when the regular system behavior is resumed after intervention or maintenance. We integrate a *conditional decoder* for the TSF model—the latest time window of the sensors is used as conditional input—to achieve these capabilities. The conditional decoding enables the model to respond faster when the sensor signals begin

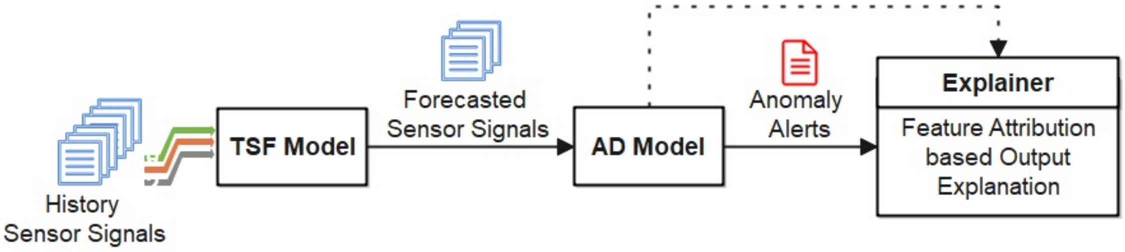


Figure 5.2: System design of the proposed AnoP system. The TSF model predicts a long sequence of signals, and the AD model produces the anomaly status of the predicted signals based on reconstruction scores. The explainer explains the detected anomalies using post-hoc feature attribution estimation (discussed in Section 4.1.2).

to evolve. We employ *dynamic decoding*—a recursive conditional decoder—to allow dynamic long-horizon forecasting. Dynamic conditional decoding is a mechanism in which earlier time slices from the model output are supplied into the decoder as conditional input to generate the subsequent output sequence. This approach has been successfully applied with S2S models in natural language processing domains, such as language translation [256, 322]. Conditional decoding without the recursive dynamic decoding has also been extended into time series data sets in recent studies [167].

Let the input time series data be $\mathrm{X}^T \in \mathbb{R}^{N_x \times T}$, where N_x is the number of input sensors with a history sequence in $t_x \in [t-T, t]$ and a length of T. The TSF model $\mathcal{S}$ predicts the sequence $\mathrm{Y}^H \in \mathbb{R}^{N_y \times H}$ with a horizon time window of $t_y \in [t+1, t+H]$ and N_y target sensors. The TSF model employs S2S encoder-decoder and the encoder $\mathcal{S}_e$ maps the input X^T into context $\mathbf{z}_e$ and state vectors $\mathbf{h}_e$ as:

$$\mathbf{z}_e, \mathbf{h}_e = \mathcal{S}_e(\mathrm{X}^T) \tag{5.1}$$

The decoder $\mathcal{S}_d$ utilizes dynamic conditional decoding that uses the context vectors $\mathbf{z}_e$ and conditional input sequences from the target sensors Y_d from the last time steps $t_d \in [t - T_d, t]$ with a size of T_d to predict the multi-timestep signals Y^H and generate decoding state $\mathbf{h}_d$ as:

$$\mathrm{Y}^H, \mathbf{h}_d = \mathcal{S}_d(\mathrm{Y}_d, \mathbf{z}_e, \mathbf{h}_d) \tag{5.2}$$

The decoder uses dynamic decoding that behaves in an autoregressive manner employing a time block-based S2S approach when inferencing long sequence horizon $H_l > H$ (see Algorithm 2). The decoder initializes its hidden states $\mathbf{h}_d$ from the encoder states—$\mathbf{h}_d = \mathbf{h}_e$—and then recursively predicts multi-timestep signal segments of the size H (line 7–11 in Algorithm 2). The latest predicted horizon Y^H is combined with the Y_d to form a new conditional input to the decoder for the subsequent forecasts (line 9).

The decoder employs a multi-attention mechanism to improve the attentiveness of the conditional inputs and leverage the forecasting accuracy (see Fig. 5.3). The model incorporates three parallel attention layers—one for the encoded context vectors $\mathbf{z}_e$, and two blocks for the conditional multivariate sensor signals Y_d on the

Algorithm 2 Multistep forecasting inference

```
procedure TimeBlockS2SMultiStepForecasting($\mathcal{S}, \mathrm{X}, \mathrm{Y}_d, H_l$)
    ▷ $F$ : is the forecasting S2S encoder-decoder model
    ▷ X : is the multivariate input times series signals with size of $N_x \times T$
    ▷ $\mathrm{Y}_d$ : is the initial decoder input from the history time window of the target signals
    ▷ $H_l$ : is the time length of the target horizon

    $H \leftarrow GetModelHorizonSize(\mathcal{S})$
    $I \leftarrow H_l/H$            ▷ the number of forecasting iterations with the basic block of $H$
    $\mathbf{z}_e, \mathbf{h}_e \leftarrow \mathcal{S}_e(\mathrm{X})$            ▷ get the learned context vectors and hidden states from the encoder
    $\mathbf{h}_d \leftarrow \mathbf{h}_e$            ▷ initializes the decoder hidden states
    $\mathrm{Y} \leftarrow [\,]$
    for $i$ in $[1, .., I]$ : do
        $\mathrm{Y}^H, \mathbf{h}_d \leftarrow \mathcal{S}_d(\mathrm{Y}_d, \mathbf{z}_e, \mathbf{h}_d)$
        $\mathrm{Y} \leftarrow Join(\mathrm{Y}, \mathrm{Y}^H)$            ▷ concatenate on the time dimension
        $\mathrm{Y}_d \leftarrow \mathrm{GetCondInput}(\mathrm{Y}^H, \mathrm{Y}_d)$            ▷ updates conditional input
    end for
    return Y
    procedure GetCondInput($\mathrm{Y}^H, \mathrm{Y}_d$)            ▷ returns decoder conditional input segment
        $H \leftarrow LENGTH(\mathrm{Y}^H)$
        $T_d \leftarrow LENGTH(\mathrm{Y}_d)$
        if $H \leq T_d$ then
            $\mathrm{Y}_d \leftarrow Join(\mathrm{Y}_d\{t \in [H, T_d]\}, \mathrm{Y}^H)$            ▷ update the latest $H$ steps of $\mathrm{Y}_d$ from $\mathrm{Y}^H$
        else
            $\mathrm{Y}_d \leftarrow \mathrm{Y}^H\{t \in [H - T_d, H]\}$            ▷ get the latest $T_d$ steps from the $\mathrm{Y}^H$
        end if
        return $\mathrm{Y}_d$
    end procedure
end procedure
```

feature quantity and time dimensions, respectively, as:

$$\begin{aligned}\psi_{\mathbf{z}_e} &= \mathrm{softmax}(\mathbf{z}_e)\\ \psi_{\mathrm{Y}_d^f} &= \mathrm{softmax}(\mathrm{Y}_d^t)\\ \psi_{\mathrm{Y}_d^f} &= \mathrm{softmax}(\mathrm{Y}_d^f)\end{aligned} \tag{5.3}$$

where the $\psi_{\mathbf{z}_e}$ is the attention scores on the learned encoder context vectors $\mathbf{z}_e$, and $\psi_{\mathrm{Y}_d^t}$ and $\psi_{\mathrm{Y}_d^f}$ are attention scores of the decoder conditional input Y_d on its temporal and feature dimensions, respectively. Attention scores are concatenated to form predictor features for the multi-timestep forecasting as:

$$\psi = [\psi_{\mathbf{z}_e} || \psi_{\mathrm{Y}_d^t} || \psi_{\mathrm{Y}_d^f}] \tag{5.4}$$

The TSF S2S model consists of residual dilated convolutional and GRU networks with attention (see Fig. 5.3). Multiple convolutional layers are stacked in the network to achieve temporal causation learning with increasing dilation size. The increasing dilation along subsequent layers expands the receptive field of the convolution operation in the time data [270, 358]. We ameliorate the model with time dimension reduction to mitigate the performance degradation for long input sequences through multilayer pooling. Residual skip connections are added in the convolutional network to enhance training with deeper layers. The decoder utilizes attention networks that take decoding inputs from the encoded latent features and conditional signals. The remaining sections of the decoder consist of blocks similar to the encoder but in reverse order with deconvolution. It employs a final

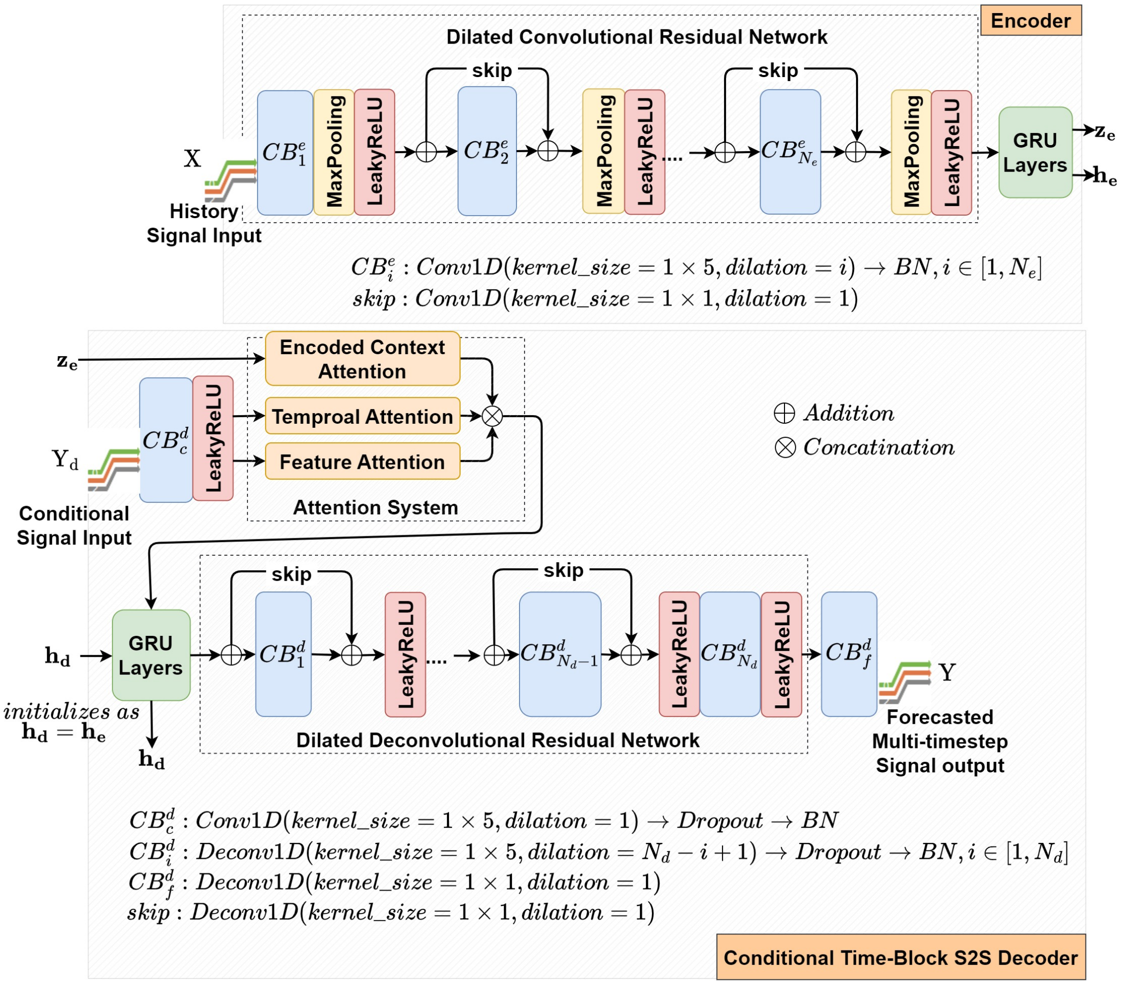

Encoder
Dilated Convolutional Residual Network
skip
skip
X
History Signal Input
CB^e_1
MaxPooling
LeakyReLU
CB^e_2
MaxPooling
LeakyReLU
....
$CB^e_{N_e}$
MaxPooling
LeakyReLU
GRU Layers
$\mathbf{z_e}$
$\mathbf{h_e}$
$CB^e_i : Conv1D(kernel\_size = 1 \times 5, dilation = i) \rightarrow BN, i \in [1, N_e]$
$skip : Conv1D(kernel\_size = 1 \times 1, dilation = 1)$
$\mathbf{z_e}$
Encoded Context Attention
Temproal Attention
Feature Attention
Attention System
$\mathrm{Y_d}$
CB^d_c
LeakyReLU
Conditional Signal Input
$\oplus$ *Addition*
$\otimes$ *Concatination*
$\mathbf{h_d}$
GRU Layers
initializes as
$\mathbf{h_d} = \mathbf{h_e}$
$\mathbf{h_d}$
skip
skip
CB^d_1
LeakyReLU
....
$CB^d_{N_d-1}$
LeakyReLU
$CB^d_{N_d}$
LeakyReLU
CB^d_f
Dilated Deconvolutional Residual Network
Y
Forecasted Multi-timestep Signal output
$CB^d_c : Conv1D(kernel\_size = 1 \times 5, dilation = 1) \rightarrow Dropout \rightarrow BN$
$CB^d_i : Deconv1D(kernel\_size = 1 \times 5, dilation = N_d - i + 1) \rightarrow Dropout \rightarrow BN, i \in [1, N_d]$
$CB^d_f : Deconv1D(kernel\_size = 1 \times 1, dilation = 1)$
$skip : Deconv1D(kernel\_size = 1 \times 1, dilation = 1)$
Conditional Time-Block S2S Decoder

Figure 5.3: TSF model: the architecture of the multi-timestep forecasting S2S encoder-decoder model. The residual block consists of multi-layered dilated CNN while the RNN blocks contain two GRU layers—encoder: 16 → 16, and decoder: 16 → 256. The convolutional blocks consist of dilated 1DCNN with 256 kernels, and the CB^d_c and CB^d_f use 16 and $N_y = 28$ kernels, respectively, for fast localized feature extraction, BN for network regularization and faster convergence, LeakyReLU for non-linear activation, and MaxPooling1D for time translation insensitive features retrieval. The decoder attention networks employ softmax. We incorporate a dropout with a drop rate of 20% for training regularization. Temporal causal learning through the CNN layers with varying sizes of dilation and the GRU layers.

deconvolution layer with a unit kernel size for output stabilization. The number of convolutional blocks on the encoder and decoder may differ; the purpose of the encoder is to extract relevant context from the history time window, whereas the decoder is to predict the signals in the horizon time window. The conditional input signals to the decoder pass through a convolutional embedding block—to extract relevant temporal features—before the attention network. The attention network in our model is not followed with a fully-connected (FC) layer to reduce model complexity—unlike previous studies [167, 172]; it is directly connected to the GRU networks, and the input weights of the first GRU layer can provide functionality similar to FC.

Multivariate Time Series Anomaly Detection Model: We employ the AD model proposed in Section 4.1.2. The AD model employs VAE $\mathcal{F}_{\theta,\omega}$ that attempts to reconstruct $\bar{\mathrm{X}}^T$ from a multivariate input data $\mathrm{X}^T \in \mathbb{R}^{N\times T}$ with N sensors on a time sequence $t_x \in [t-T, t]$. The encoder of the model provides normally distributed low-dimensional latent signals $\mathbf{z} \in \mathbb{R}^{N_z\times 1}$ as:

$$\bar{\mathrm{X}}^T = \mathcal{F}_{\theta,\omega}(\mathrm{X}) = D_\omega(E_\theta(\mathrm{X}^T)) \tag{5.5}$$

The model estimates anomaly scores from the signal reconstruction errors. The reconstruction error $\mathbf{e}$ for each univariate sensor is calculated based on MAE as:

$$\mathbf{e}_i(t) = \frac{1}{T}\sum_{t'=t-T}^{t} |\mathbf{x}_i(t') - \bar{\mathbf{x}}_i(t')| \tag{5.6}$$

where $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$ are the input and reconstructed univariate data for the s^{th} sensor. We standardize the reconstruction error as follows:

$$\mathbf{s}_i = (\mathbf{e}_i - \mu_i)/\sigma_i \tag{5.7}$$

where $\mu_i = \mathbb{E}[e_i]$ and $\sigma_i = \sqrt{\mathbb{E}[(e_i - \mu_i)^2]}$ are the mean and standard deviation, respectively, calculated on the training dataset. The reconstruction error is standardized to compensate for the reconstruction performance variations across the sensors. The multidimensional reconstruction error is finally converted into an anomaly score using Mahalanobis distance (d_{MD}) [350]—the multidimensional distance between a vector and a distribution as:

$$d_{MD} = \sqrt{(\mathbf{s}_j - \mathbf{s}_\mu)^T \cdot C^{-1} \cdot (\mathbf{s}_j - \mathbf{s}_\mu)} \tag{5.8}$$

where the vector s_j is the multivariate anomaly score of the j^{th} observation, the vector $\mathbf{s}_\mu = \mu_1, \mu_2, \ldots, \mu_N$ contains the mean values of the univariate anomaly scores estimated on the training dataset, and C^{-1} is the inverse covariance matrix of S. We generate the MD anomaly flags through $A_{MD} = d_{MD} > K_{MD}\mu_{MD}$, where K_{MD} and μ_{MD} are the detection sensitivity and $\mu_{MD} = \mathbb{E}[d_{MD}]$ on the training dataset, respectively.

The AD model employs 1DCNN and GRU networks—adopted from our previous work on MTS AD in [41] (see Fig. 4.1 in Section 4.1). The CNN consists of three blocks of 1DCNN (64 kernels, *kernel_ size*=1×3) with MaxPooling1D and MaxUnpooling1D with a stride size of 2 for the encoder and decoder, respectively. The RNN consists of two GRU layers—encoder *hidden_ size*: $16 \rightarrow 4$, and decoder: $4 \rightarrow 16$—and FC layers implement the variational network.

Model Training: We trained the TSF and AD models of the AnoP system separately since the models require different training data sets. The AD needs training data with healthy instances or low anomaly contamination, while the TSF requires substantial predictable anomalies in its training dataset. Obtaining clean data of healthy instances in the training data is one of the main challenges of semi-supervised learning of AD models [99]. We cleaned the potential outliers from each univariate sensor data in the training set using a state-of-the-art time series outlier detection algorithm—saliency residual [41,95]. The TSF S2S model was trained on the dataset with anomalous patterns to leverage its capability to forecast anomaly signals from early signs. The modeling approach is fully unsupervised and does not require any labeling. The model may struggle to learn the anomaly signals due to the class imbalance since anomalies are rare instances. We attempted to mitigate the challenge with the support of the AD model. We selected the data sources—the RBXes—that have a significant number of outliers—potential anomalies—spanning substantial periods on the sensor data.

We trained the models with *Adam* optimizer using a super-convergence *one-cyclic* learning rate scheduling mechanism [356]. We have applied a *cyclic annealing* method [359] when KL divergence loss regularizes the training cost function to mitigate KL loss vanishing—latent squashing—for the VAE of the AD model.

5.1.3 Experimental Results and Discussion

We trained the TSF and AD models of the AnoP on 28 sensors per RBX. We utilized the same sensors for the input and target—i.e., $N_x = N_y = 28$. We trained the TSF S2S and the AD models on two-month data (10–11/2018) from six RBXes (HEM01, HEM04, HEM17, HEP14, HEP15, and HEP18) and one-month data (10/2018) from four stable RBXes (HEM01, HEM07, HEM17, and HEP11), respectively; we developed the models with PyTorch and trained them up to 5000 iterations. We evaluated the performance on the 25/09–03/12/2018 date range on thirty-four RBX systems.

The TSF model employs $N_{cd}^{e} = 2$ and $N_{cd}^{d} = 4$ causal residual convolution blocks for the encoder and decoder networks, respectively. The model operates on a $T = 120$ hours—equivalent to 5 days—sliding history time-window with prediction horizon sizes of $H = [24, 168]$ hours—1 to 7 days; the base forecasting horizon $H = 24$ hours. The conditional decoder of the model uses the last $T_d = 24$ hours from the history time window for the target sensors.

The AD model predicts anomalies on the 24-hour sliding window. We heuristically set $\alpha_{MD} = 10$ to estimate the reconstruction-based AD decision thresholds. We have compared the AP performance of the AnoP with the benchmark CGVAE AD model. The benchmark model is the same as the AD model of the AnoP, except it detects the anomaly from the raw sensor signals in contrast to the AnoP that the AD model detects anomalies from the forecasted signals. We compare AnoP prediction performance with the anomaly flags generated by the AD model on measured raw signals—non-forecasted—because of the lack of labeled anomaly data.

Multi-timestep Forecasting Model Evaluation: We have validated the efficacy on multiple extended horizon sizes—24 to 168 samples (see Table 5.1). The results demonstrate that the model forecasted long horizons with slight performance degradation through the time block S2S mechanism.

Table 5.1: Forecasting performance on MTS data, averaged over all the RBX systems.

Horizon (H)	**24h**	**48h**	**72h**	**96h**	**120h**	**144h**	**168h**
MAE	0.418	0.430	0.444	0.464	0.473	0.503	0.529
MSE	1.392	1.416	1.465	1.515	1.558	1.635	1.705

MAE is mean absolute error, and MSE is mean squared error.

Fig. 5.4 illustrates the forecasting capability of the proposed attention mechanism with conditional decoding compared to a conditional decoder without attention. The MAE and MSE scores improve substantially by 10–15% and 22–28%, respectively. Fig. 5.5 portrays an ablation study on the TSF model to demonstrate the contribution of the building blocks of the temporal model—the attention, conditional decoding, and convolution layers.

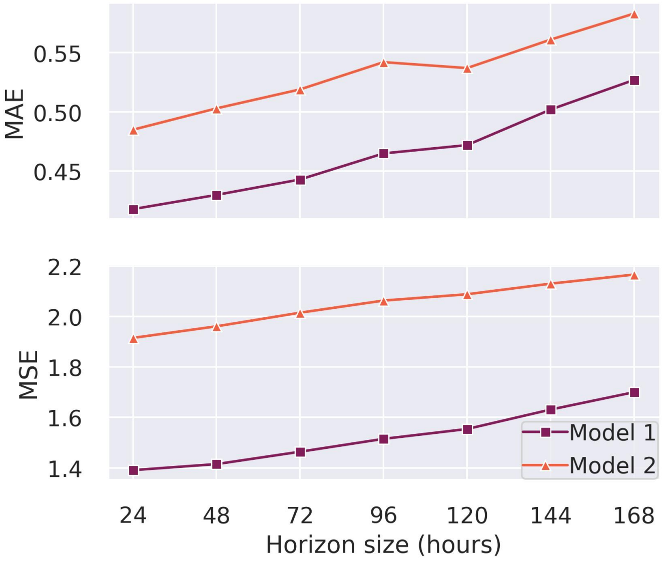
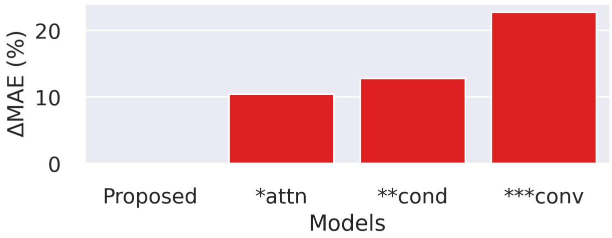


Figure 5.4: Multivariate time series forecasting performance comparison between different model configurations: *Model 1*: the proposed attention-based conditional decoder, and *Model 2*: conditional decoder without attention.

Anomaly Prediction Performance: We define an anomaly as an outlier that deviates from the expected nominal characteristics due to the lack of annotated data;



* is the number of excluded blocks from the proposed TSF model

Figure 5.5: Ablation performance evaluation of the TSF model at $H = 24$ hours. The MAE score difference in percentage is given relative to the proposed model. **attn—w/o attention, **cond—w/o conditional decoding*, and ****conv—w/o convolution layers.*

not all anomaly flags thus indicate failure in the detector. We validated the efficacy of the AD model compared to flags generated from error-counter variables of the HCAL discussed in the previous Section 4.1.3 [41]. However, we found those counter quantities are less convenient for AP evaluation as they are ineffective in capturing most of the gradual system deterioration anomalies [41]. We thus generated reference anomaly labels from the AD model flags on the raw data—not forecasted—to evaluate our AP system. The AD model has generated an average of approximately 160 anomalous data flags per RBX on raw data from the monitored sensors over ten weeks (see Fig. 5.6). A few RBXes have generated more flags due to higher variability on the readings from the 1V2_CURRENT sensor on the slave control card of ngCCM (further discussion at the end of this section).

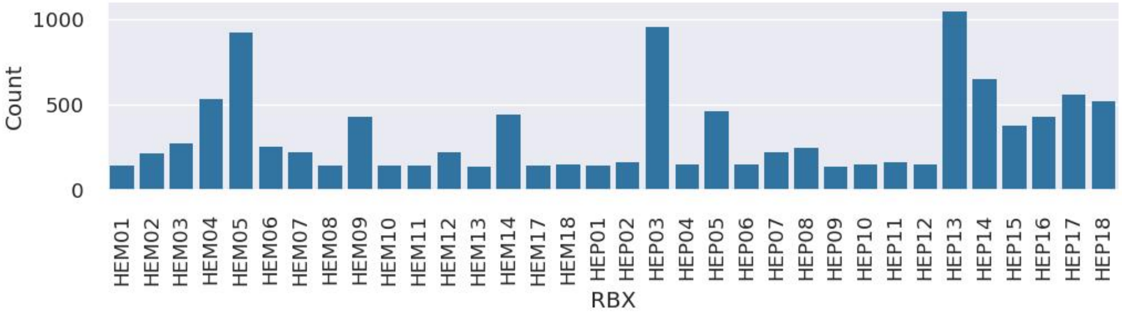
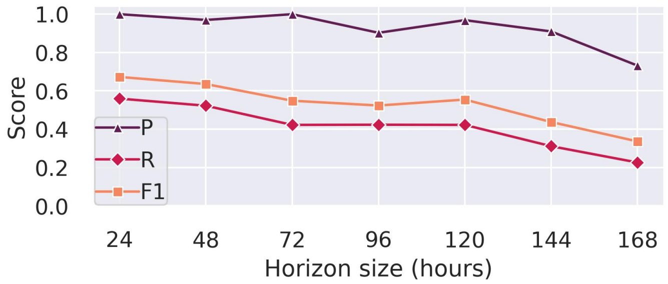


Figure 5.6: Number of anomaly data points—detected by the CGVAE AD model—used as reference flags for evaluating the AP system. The high number of anomalies in the HEM05, HEP03, HEP05, HEP13, and HEP14 is due to the noisy behavior of the 1V2_CURRENT sensor of the ngCCM slave control card.

Fig. 5.7 and 5.8 portray the classification performance on prediction accuracy of the proposed AnoP system. The AnoP has predicted long horizon anomalies with high precision—demonstrating the robustness of the proposed system in avoiding false flags (see Fig. 5.7). The recall is just below 0.60 despite the promising performance in precision; this is caused by missed anomalies arising from unpredictable transient behavior—additive noise is a prime cause of transient anomalies.

Our models have revealed a noisy behavior of the 1V2_CURRENT sensor of the ngCCM slave control card of some RBXes. Fig. 5.9 illustrates an example of the sensor's behavior and AnoP predictions. The AD model has generated substantial anomaly flags for those particular RBXes (see Fig. 5.6), but the AnoP has struggled to achieve good anomaly forecast—low recall—due to a lack of learnable causal patterns (see Fig. 5.8). This is the first observation of the phenomenon in the HCAL, even though the behavior is not entirely unexpected. The slave control cards can be noisier due to the mounted FPGA's attempt to lock onto a non-existent incoming data stream since the cards do not maintain the backend communication link. This behavior does not impact operation, but monitoring its status would provide relevant information when the decision to switch the master ngCCM control card is made.



* P- precision, R- recall, F_1- f1-score

Figure 5.7: AP performance of the AnoP compared to the CGVAE AD across different time horizons.

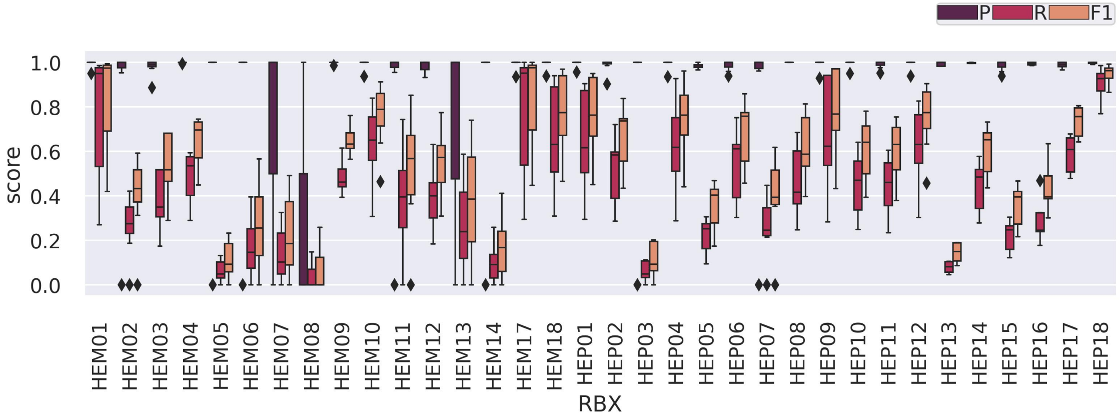


Figure 5.8: AP performance distribution of AnoP on multiple horizons across the RBX systems. The lower performance in some RBXes is because of the additive transient anomalies and noisy slave control card sensors. A missing sensor in HEM08 has impacted the prediction accuracy; we imputed the data with a constant nominal value.

Persistent anomalies are often indicators of severe problems, and Fig. 5.10 portrays successfully forecasted persistent outliers in the current and voltage sensors of the RBXes during 28/10–03/11/2018. There had been LHC maintenance tasks—the machine development and technical stop tasks—during that time; the tasks are

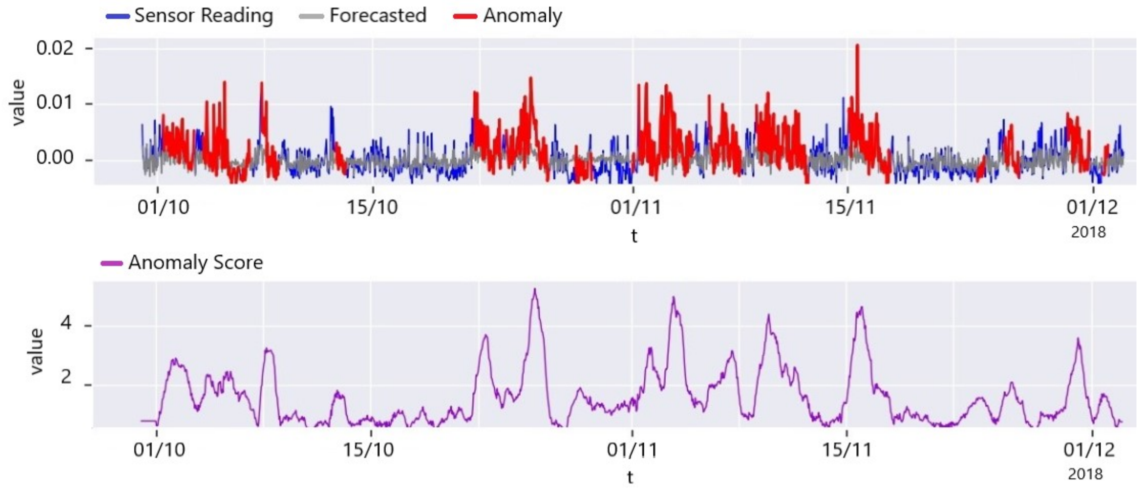


Figure 5.9: Anomaly prediction on 1V2_CURRENT sensor of the ngCCM slave control card of RBX HEP05. The sensor is noisier in some RBXes; five times stronger noise-like fluctuation—around 0.01A—is observed compared to its corresponding master card—0.002A—and slave card of the other RBXes. The y-axis value is the normalized reading after subtracting the nominal value.

planned in the LHC operation schedule to study and optimize the long-term performance of the LHC. Further investigations—following our finding—indicate that the LHC tasks had unexpectedly affected the low-voltage supply of the RBX. The impact was within tolerance and did not negatively impact the performance of the HCAL; but, the knowledge from the AnoP allows the HCAL team to better prepare for LHC interventions in the future.

5.1.4 Summary

Predictive maintenance aims to achieve versatile leverages in significantly cutting maintenance costs and downtimes as the pillar application of the fourth industrial revolution. We have demonstrated the efficacy of the anomaly prediction approach—AnoP—through unsupervised end-to-end long-time series forecasting and anomaly detection mechanisms on multivariate time series data. The evaluation of the Hadron Calorimeter diagnostic sensor data sets has unveiled that anomalies that persevere for a certain period can be forecasted from early indications. The AnoP system will enable prognostics and predictive maintenance in the HCAL during the LHC high-energy Run-3 collision experiment. Our proposed methods of the AnoP are generic enough to be applied with less effort for predictive maintenance applications in other domains with time series data.

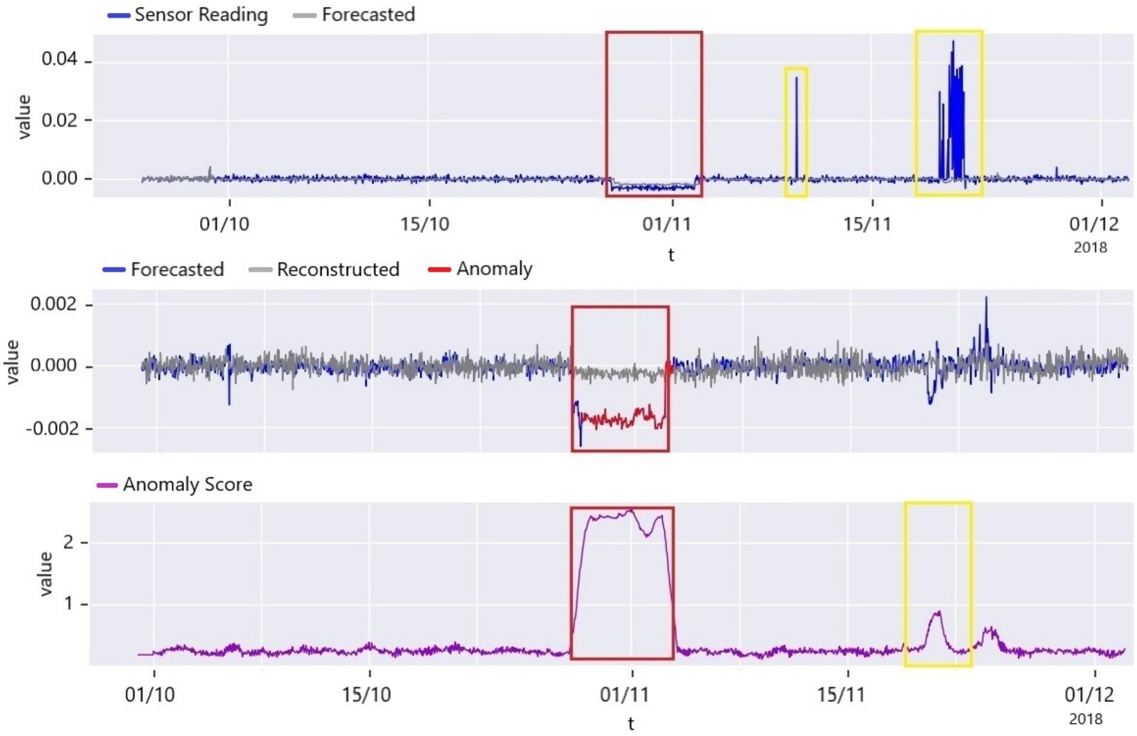


Figure 5.10: Forecasting persistent and transient anomalies on the 1V2_CURRENT sensor of the master control card of the HEP03: (top) the forecasted signal with $H = 24h$ compared to the ground truth, (middle) reconstruction using the AD AE from the forecasted signal, and (bottom) the predicted anomaly score. The red boxes show accurately predicted persistent anomalies, and the yellow boxes enclose the transient and spike outliers that are challenging to forecast.

Chapter 6

Anomaly Diagnostics

This chapter presents our studies on causal discovery and analysis of anomalies in multivariate time series data.

6.1 Lightweight Mechanism for Multi-System Interconnection and Divergence Discovery

Multivariate sensors are widely employed for monitoring large systems, and identifying outlier behavior among sensors and subsystems is essential for discovering faults and facilitating diagnostics. At the same time, exploring large systems with numerous multivariate data sets is challenging. We present in this study a lightweight mechanism for analyzing the interconnections among multiple sensors and systems. The interconnection exploration aims to identify abnormal variables and systems from multi-systems monitored with multivariate sensors. The approach presents a multivariate analysis technique that first estimates the interconnection heatmaps among the sensors for each system and then applies several information retrieval algorithms—interpretation—to provide relevant interconnection and discrepancy details. The mechanism generates clusters and association links—allowing us to identify abnormal systems and the root cause of the diverging behavior, the faulty sensors. Our experiment on the readout systems of the Hadron Calorimeter demonstrates the effectiveness of the proposed method in clustering sensors and systems consistent with the actual expected configuration of the detector, while systems with unusual sensor readings form divergent clusters.

Multivariate analysis (MVA) is a statistical method utilized for analyzing data involving multiple variables; it considers multiple factors or variables to provide more accurate insights into the level of influence on variability—summarize the relationships into fewer statistics while preserving the main facets of the relationships. Various domains utilize MVA techniques for data reduction, grouping, clustering, dependency analysis, and hypothesis testing due to the multivariate nature of real-world problems [360]. To effectively utilize MVA techniques, it is essential to comprehend the various aspects of the techniques regarding the problems they are suitable for, their objectives, the necessary data structure, and the underlying mathematical model. Several methods can be used for multivariate analysis—including principal component analysis (PCA), factor analysis (FA), canonical correlation analysis (CCA), structural equation modeling (SEM), and multidimensional scaling (MDS). The best method to use depends on the type of data and the problem to be solved. PCA and FA are dimensionality reduction methods for multivariate data. FA uses PCA for dimension reduction and common factor analysis to group highly correlated variables. SEM generates linear prediction model equations for each dependent variable from a set of independent variables. CCA determines a set of canonical variates—orthogonal linear combinations of the variables within each set—that best explain the variability within and between multivariate sets [361]. MDS creates a representation map displaying the relative positions of several objects in lower dimensions from similarity distance matrices [362]. There are nonlinear manifold dimensionality reduction techniques that aim to preserve the local structure of data, such as t-stochastic neighbor embedding (t-SNE) [16]; and uniform manifold approximation and projection (UMAP) [18]. The existing MVA methods struggle to handle multi-sources with multivariate data or provide interpretability of their results. In this study, we focus on interpretable cluster analysis for interconnection and discrepancy discovery of multi-systems monitored with multivariate sensors. Clustering analysis deals with unsupervised grouping of similar objects based on measured characteristics. It often accompanies the abovementioned dimension reduction methods to group objects into different classes automatically—adaptive to changes and identifies valuable features that differentiate different groups. Additionally, clustering enables outlier detection applications where divergent behavior is considered unusual.

The HCAL consists of four subdetectors that capture the collision physics particles from different positions of the calorimeter [3]. Each subdetector is composed of *readout boxes* (RBXes)—incorporating frontend data acquisition electronics components. The RBXes per a subdetector share similar operations, technology, and configurations. The RBX electronic systems are currently monitored through several multivariate diagnostics sensor variables [8]. Most of the sensors are employed in retrospective monitoring and debugging, which primarily rely on simple statistical analysis and visual inspection of a large and diverse set of signals. Previous machine learning efforts on anomaly detection (AD) monitor the RBX subsystems using pre-trained deep learning models on a specific set of sensor variables [41, 70]. However, these models have limited flexibility and computational overhead, as they require offline training.

This study investigates online discrepancy discovery in HCAL subdetectors' multi-RBX system configuration without prior training by analyzing deviations in sensor interconnection behavior. The aim is to provide a lightweight, generic, scalable, dynamic, and online fault detection and diagnostic system through multivariate analysis. Our experiment on the readout systems of the HCAL validates the effectiveness of the proposed method in clustering sensors and systems consistent with the existing expected configuration of the detector. Systems with unusual sensor readings form divergent clusters, and our approach has identified the potential root cause of the divergence with visual illustrations and quantitative scores. Our approach shares similarities with CCA [361] and MDS [361] as multivariate statistical analysis for the discovery and quantification of associations among sets of multivariate. Most of the existing techniques—including CCA and MDS—provide tools to display low data dimensions and largely leave the interpretation of the data to the human observer [16]; CCA runs on a pair of sets with multivariate, and MDS is a descriptive technique without statistical inference. The methods also require tabular data that incomplete observations may impact their performance—for instance, real-world sensor data reading exhibits several measurement issues, such as missing data. Our approach tolerates measurement quality issues if the sensor interconnections within a given system are unaffected.

6.1.1 Dataset Description

We utilized sensor data from the RM of the HE, and each RM has twelve diagnostic sensors—four from the SiPM control card and eight from the four readout QIE cards (see Table 6.1). The dataset was obtained from the HCAL software monitoring system (ngCCM server) from 01/08/2022 to 30/11/2022. The monitoring sensor data comprises four-month data of 20.7M samples—around 12K per sensor per RM. The dataset contains irregularly sampled and considerably sparse data—a few a-minute interval samples are logged every eight and two hours for the SiPM control card and QIE card sensors, respectively. We downsampled the HE-RM-1 dataset—RM-1 data from all the 36 RBXes—to hourly intervals—and used a smoothed version after transient spike removal and interpolating short gaps. Our main objective for the interconnection analysis is to identify any time-persistent discrepancies in multivariate sensor data among the RBXes.

Table 6.1: Data variables description.

No.	**Notation**	**Variable Name**	**Remark**
1	SPV	PELTIER_VOLTAGE_F	Voltage of the SiPM Peltier temperature controller
2	SPC	PELTIER_CURRENT_F	Current of the SiPM Peltier temperature controller
3	SRT	RTDTEMPERATURE_F	SiPM CC temperature averaged over 50 samples
4	SCH	HUMIDITY_F	SiPM CC humidity
5	Q1H	1-B-SHT_RH_F	QIECARD 1 humidity
6	Q2H	2-B-SHT_RH_F	QIECARD 2 humidity
7	Q3H	3-B-SHT_RH_F	QIECARD 3 humidity
8	Q4H	4-B-SHT_RH_F	QIECARD 4 humidity
9	Q1T	1-B-SHT_TEMP_F	QIECARD 1 temperature
10	Q2T	2-B-SHT_TEMP_F	QIECARD 2 temperature
11	Q3T	3-B-SHT_TEMP_F	QIECARD 3 temperature
12	Q4T	4-B-SHT_TEMP_F	QIECARD 4 temperature

6.1.2 Methodology

The interconnection discrepancy exploration approach involves four main stages (see Fig. 6.1): 1) sensor similarity map generation for each system, 2) system similarity distance map estimation using sensor interconnection maps, 3) system and sensor clustering analysis, and 4) divergence root-cause discovery.

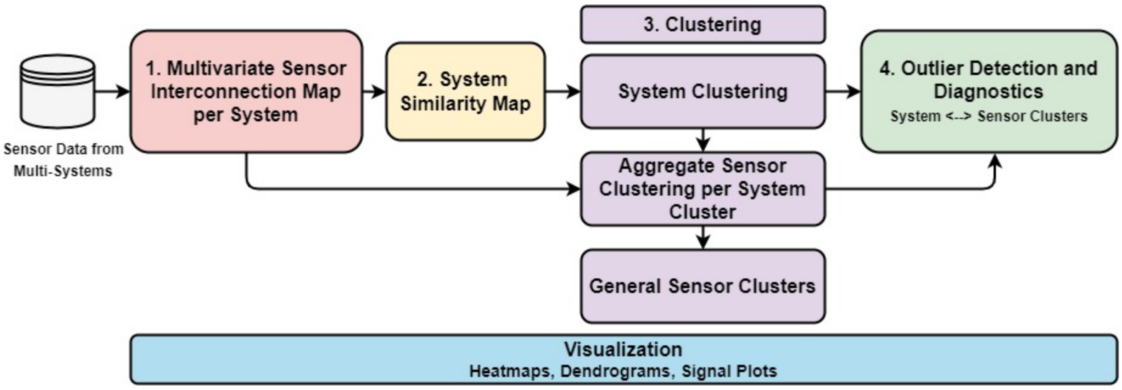

Sensor Data from Multi-Systems
1. Multivariate Sensor Interconnection Map per System
2. System Similarity Map
3. Clustering
System Clustering
Aggregate Sensor Clustering per System Cluster
General Sensor Clusters
4. Outlier Detection and Diagnostics
System <--> Sensor Clusters
Visualization
Heatmaps, Dendrograms, Signal Plots

Figure 6.1: Our lightweight multi-systems multivariate interconnection divergence discovery approach.

- *Sensor similarity map generation* (I): We estimate pairwise interconnection I among the multivariate sensors X of a given system using Γ function as:

$$I_k[i,j] = \Gamma(X_k[:,i], X_k[:,j]),\ i \neq j,\ (i,j) \in S,\ k \in M \tag{6.1}$$

where $I_k \in \mathbb{R}^{N_s \times N_s}$ is the pairwise interconnection matrix, and $X_k \in \mathbb{R}^{T \times N_s}$ is sensor data—with N_s sensors and T data samples—of the k^{th} system in the system set M. The Γ is a similarity measurement between the i^{th} and j^{th} sensors in the sensor set S. The I is the interconnection map, a two-dimensional matrix holding the pairwise score between the sensors. We employ a bivariate *Pearson's correlation* [363] for the Γ for its fast computation and decent accuracy; the high data sampling interval—time delay effect is lessened in our use-case data. Our data sets fairly adhere to normal distributions, and we have cleaned noisy transient outliers to partially address the potential constraints of ρ (see Section 6.1.1). The Pearson's correlation measures the linear correlation between two variables; it is the ratio between the covariance and the product of their standard deviations:

$$\rho(\mathbf{x_1}, \mathbf{x_2}) = \frac{\sum_{i=1}^{n} (\mathbf{x_1} - \bar{\mathbf{x}}_\mathbf{1})(\mathbf{x_2} - \bar{\mathbf{x}}_\mathbf{2})}{\sqrt{\sum_{i=1}^{n} (\mathbf{x_1} - \bar{\mathbf{x}}_\mathbf{1})^2}\sqrt{\sum_{i=1}^{n} (\mathbf{x_2} - \bar{\mathbf{x}}_\mathbf{2})^2}} \tag{6.2}$$

where ρ is the correlation coefficient between sensor reading vector $\mathbf{x_1}$ and $\mathbf{x_2}$. The ρ is a normalized covariance measurement—$\rho \in [-1, 1]$; positive values for a simultaneous increase and negative values for otherwise, small $|\rho|$ indicates a weak correlation, and zero implies no linear correlation. Other methods like *dynamic time wrapping* and *Granger's* may provide enhanced accuracy on time series data with higher computation overhead.

- *System similarity map estimation* (D): We utilize normalized pairwise Euclidean distance Φ among the sensor similarity maps of the systems as:

$$D[k,h] = \Phi(I_k, I_h),\ k \neq h,\ (k,h) \in M \tag{6.3}$$

where $D[k,h]$ is the similarity distance score between the I of k^{th} and h^{th} systems, and the $D_k \in \mathbb{R}^{1\times N_m}$ is the similarity distance vector of the k^{th} system—holding the pairwise score between the system k to all other systems—$m \in M,\ m \neq k$. The Φ is calculated as:

$$\Phi(I_k, I_h) = \frac{1}{N_s}\sqrt{\sum_{i\in S}^{N_s}\sum_{j\in S}^{N_s}(I_k[i,j] - I_h[i,j])^2} \tag{6.4}$$

where $\frac{1}{N_s}$ is the score normalizing factor.

- *System and sensor clustering analysis* (C and ξ): We infer the system and sensor clustering through hierarchical agglomerative clustering Θ with time complexity of $\mathcal{O}(n^2)$, where n is the number of observations [17]. The clustering link distance $C^m \in \mathbb{R}^{N_m\times N_m}$ of the systems is calculated on the D using Euclidean distance Φ and clustering optimization through nearest-neighbors chain algorithm [17]:

$$C^m = \Theta(\Phi, D) \tag{6.5}$$

The system clusters are generated by applying a threshold α^m on the C^m:

$$L^m[k,h] = C^m[k,h] < \alpha^m,\ k \neq h,\ (k,h) \in M \tag{6.6}$$

where $L^m \in \mathbb{Z}^{N_m\times N_m}$ holds the binary cluster links—with active edges for all the systems belonging to the g^{th} cluster ξ_g^m—$\{k,h\} \in \xi_g^m : L^m[k,h] = 1$. This generates N_ξ system clusters.

We calculate sensor clustering using a multi-step approach: 1) we first estimate the sensor interconnection map $I_g^m \in \mathbb{R}^{N_s\times N_s}$ per each system cluster ξ_g^m, and 2) we generate sensor cluster links $L_g^s \in \mathbb{R}^{N_s\times N_s}$ from the I_g^m for each ξ_g^m. Since each system cluster in ξ^m represents distinct sensor interconnection characteristics, I_g^m is calculated as average sensor interconnection maps from all the systems in the cluster:

$$I_g^m[i,j] = \frac{1}{N_g}\sum_{k\in\xi_g^m} I_k[i,j],\ (i,j) \in S \tag{6.7}$$

where $I_g^m[i,j]$ is the average sensor interconnection score between the i^{th} and j^{th} for the ξ_g^m system cluster. The I_k and N_g correspond to the sensor interconnection map of the member system $k \in \xi_g^m$ and the number of systems in ξ_g^m, respectively. We measure the pairwise sensor cluster distance score $C_g^m \in \mathbb{R}^{N_s\times N_s}$ per the given system cluster ξ_g^m using Φ as:

$$C_g^s = \Theta(\Phi, I_g^m) \tag{6.8}$$

The sensor clustering links per system cluster are thus generated similarly by applying a threshold α^s on the C_g^m:

$$L_g^s[i,j] = C_g^s[i,j] < \alpha^s,\ (i,j) \in S \tag{6.9}$$

We estimate the overall sensor interconnection map $I^s \in \mathbb{R}^{N_s \times N_s}$, the interconnection clustering distance $C^s \in \mathbb{R}^{N_s \times N_s}$, and cluster links $L^s \in \mathbb{Z}^{N_s \times N_s}$ by averaging all the system clusters:

$$\begin{aligned} I^s[i,j] &= \frac{1}{N_\xi} \sum_{g \in \xi^m} I_g^m[i,j] \\ C^s &= \Theta(\Phi, I^s) \\ L^s[i,j] &= C^s[i,j] < \alpha^s,\ i \neq j,\ (i,j) \in S \end{aligned} \tag{6.10}$$

where N_ξ is the number of system clusters. We characterize the discrepancy root causes—the divergent sensors responsible for system cluster splitting—and overall sensor interconnection within and across systems by inferring the sensor linkages among the system clusters from the I_g^m and I_g^s through heatmap plots and dendrogram clustering linkage diagrams.

- *Divergence root-cause discovery* (ψ and R): We discover the root-causes—sensor variables—that contributes significantly to cluster split among the systems using the aggregated difference of average interconnection maps I_g^m of the system clusters as:

$$\psi_g^m[h,i] = \Phi(I_g^m[i,:], I_h^m[i,:]),\ (g,h) \in \xi^m,\ i \in S \tag{6.11}$$

where $\psi_g^m[h,i]$ is the sensor interconnection divergence scores of the cluster g from cluster h at the i^{th} sensor variable.

$$\bar{\psi}_g^m[i] = \sum_{h \in \xi^m} \psi_g^m[h,i],\ i \in S \tag{6.12}$$

where $\bar{\psi}_g^m \in \mathbb{R}^{1 \times N_s}$ holds the sensor divergence score for the system cluster ξ_g^m. We apply a threshold α^ϕ for select significant root cause contributions R_g^m as:

$$R_g^m[i] = \bar{\psi}_g^m > \alpha^\phi \tag{6.13}$$

6.1.3 Experiment Results and Discussion

We have applied our proposed approach for online interconnection discrepancy discovery to RMs from the HE's thirty-six RBX system—each monitored by twelve sensor variables. We will discuss the results of interconnection discrepancy exploration in this section.

The heatmap plot in Fig. 6.2 and clustering diagram in Fig. 6.3 illustrate the behavioral discrepancy among the RBX systems, respectively. The similarity of the sensor pairwise correlation maps of the RBXes determines the distance; the RBXes are clustered based on their similarity in sensor interconnection behavior. The clustering indicates outlier behavior—a small number of cluster members—on the RBX systems in CL-1 (HEP07 and HEP08), and CL-5 (HEP02). We have applied dimension reduction methods, such as principal component analysis (PCA), stochastic neighbor embedding (t-SNE) [16], and uniform manifold approximation and projection for dimension reduction (UMAP) [18], to the system dissimilarity matrix D to visualize the clusters in two-dimensional (see Fig. 6.4). The figures demonstrate that reduction algorithms can capture the variations but may fall short of providing a clear split in some clusters. Identifying the divergence root causes from the dimension reduction algorithms is not straightforward and remains challenging. The following paragraph will discuss our proposed mechanisms for discovering diagnostic root causes of the estimated hierarchical clustering of the RBX systems.

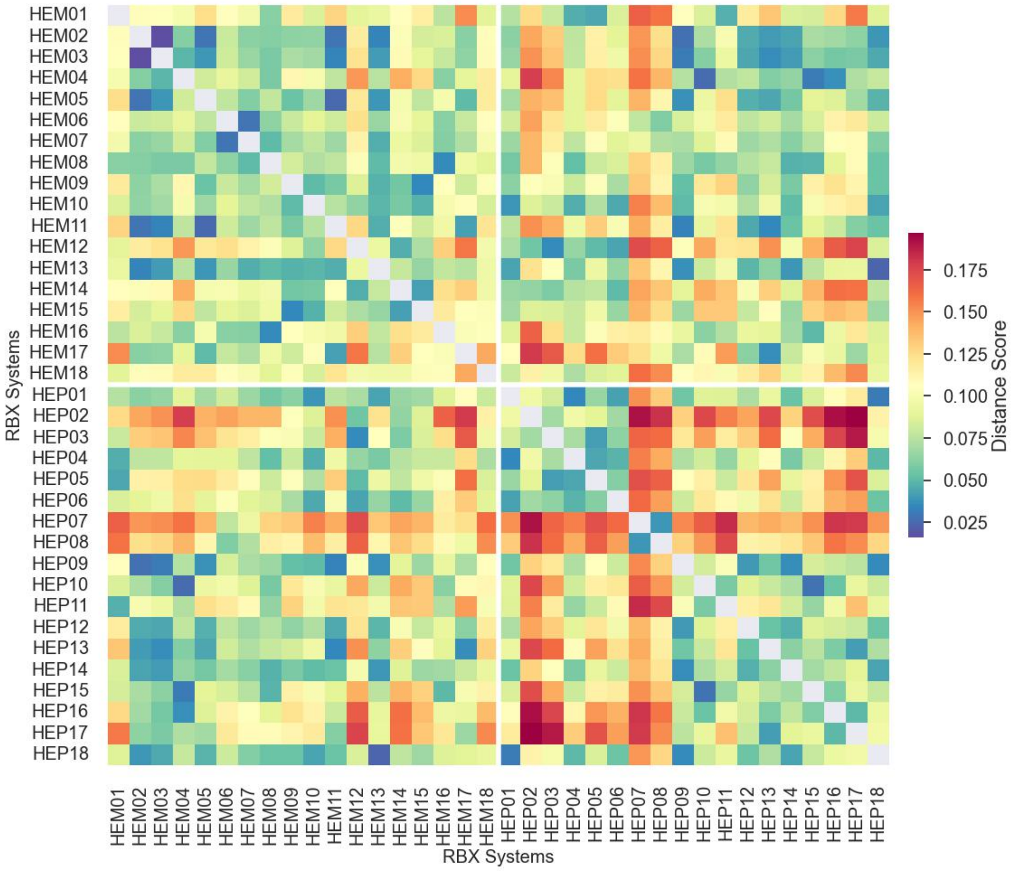
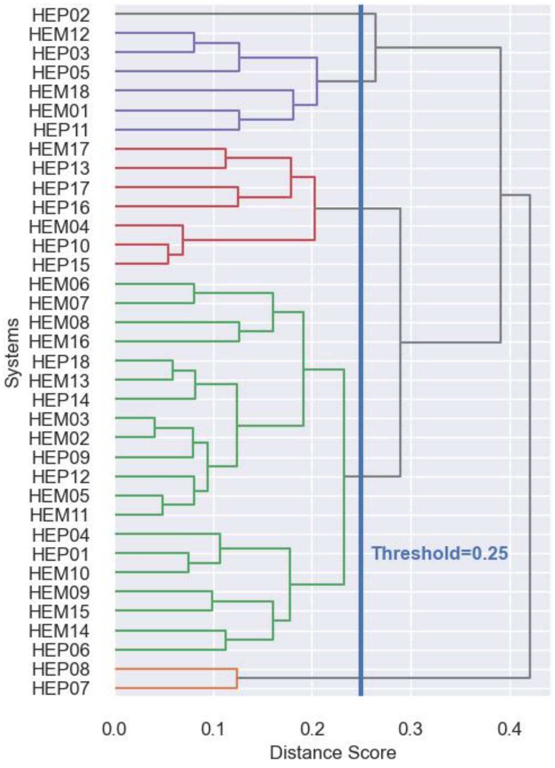


Figure 6.2: RBX-RM multi-systems dissimilarity distance heatmap among the RBXes (D). The color bar shows the score, the normalized Euclidean distance among I_ks maps of systems.

The representational sensor interconnections for the system clusters—calculated from Eqs. (6.8) to (6.9)—captures sensor interconnections variations that led to the system cluster splits. We portray the results from Fig. 6.5 to 6.7. Fig. 6.5 presents the sensor interconnections clustered dendrograms per RBX cluster—L_g^s estimated by (6.9). The dendrograms highlight differences in the interconnection of SCH and SRT sensors among RBX clusters. Fig. 6.5f depicts the average sensor



The shown scores are without scaling normalizing factor in Φ, the number systems in Eq. (6.4).

Figure 6.3: RBX system clustering (C^m), using *hierarchical clustering* [17] on D. The clustering demonstrates the similarity and divergence among the systems. The threshold at $\alpha^m = 0.25$ generates five clusters (N_ξ=5): CL-1, CL-2, CL-3, CL-4, and CL-5, where the CL-i denotes the i^{th} cluster.

interconnection aggregating the distinct characteristics across the RBX clusters using (6.10). Fig. 6.6 provides the visual diagnostics heatmap through I_g^ms to detect any discrepancies in the sensor connections across the RBX system clusters. The difference in coloring within each column box indicates which sensors per the RBX clusters behave differently, while uniform coloring indicates similarity in behavior. For example, the SCH and Q[1-4]H sensors exhibit discrepancies. Fig. 6.7 depicts the equivalent clustering dendrogram of Fig. 6.6 to capture the differences quickly besides the heatmap visualization. The figure portrays the MTS sensor interconnections across all the RBX system clusters; sensors measuring similar quantities across similar subsystems are clustered together—e.g., SiPM card: SPV, SPC, SRT, and SCH, and QIE cards: Q[1-4]T and Q[1-4]H. The behavioral similarity among the four QIE card sensors within an RBX cluster is more substantial than across RBX clusters. Divergence exceptions are SCH of the RBX CL-5. It is possible to generate different sensor interconnection cluster coloring by increasing and decreasing the α^s to capture solid and subtle differences among the RBX clusters—for instance, lowering the threshold to $\alpha^s = 0.05$ isolates the Q[1-4]H of the CL-5 (see Fig. 6.7b).

The generated general knowledge of the system and sensor interconnection from the above clustering illustrations can be summarized as: 1) the voltage and current

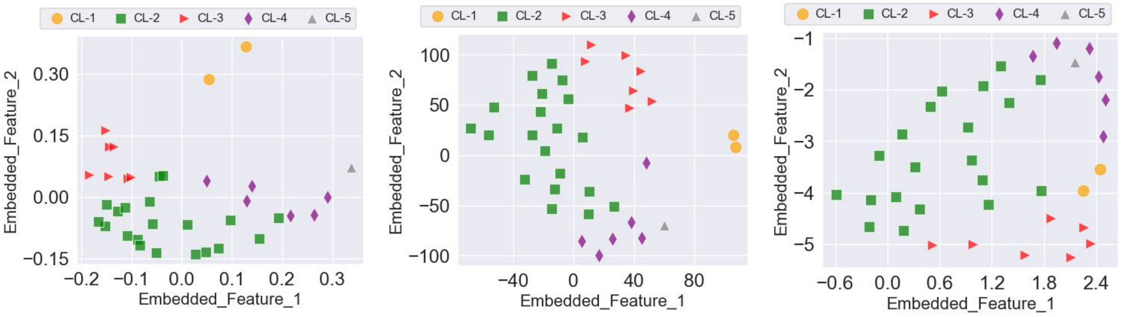
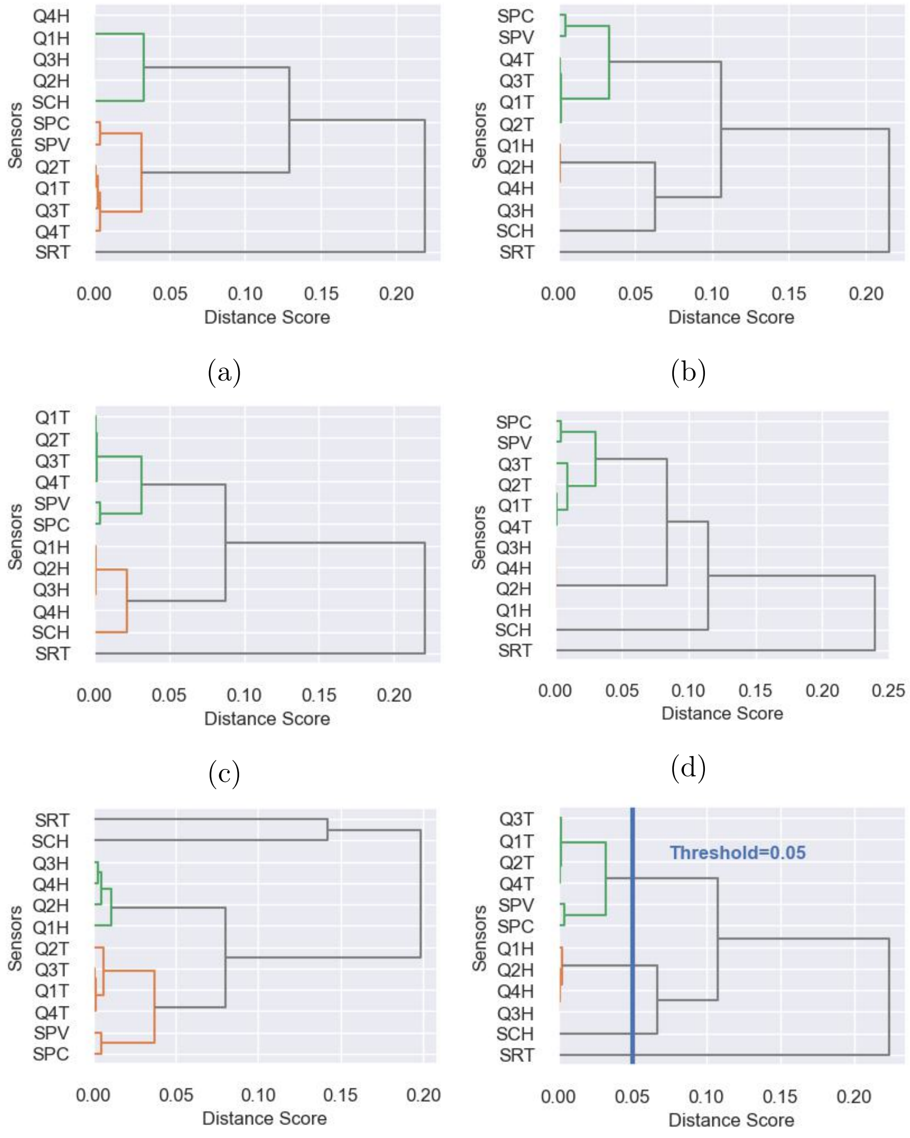


Figure 6.4: Dimension reduction on the similarity distance score D using a) PCA, b) t-SNE [16], and c) UMAP [18]. There are still overlaps among the clusters that suggest a higher number of features for dimension reduction is required.

sensors of the SiPM card—SPV and SPC—are strongly interconnected, 2) the temperature sensors (Q[1-4]T) of the QIE cards are strongly interrelated and connected to the SiPM card SPV, and SPC sensors, 3) the humidity sensors (Q[1-4]H) of the QIE cards are strongly interrelated and connected to the SCH but with weaker strength, and 4) the SRT sensor has distinct and isolated behavior. The association between the QIE card sensors for humidity (Q[1-4]H) and temperature (Q[1-4]T) weakens—distance increases—in the RBX CL-4. The difference in the sampling method used for the sensors may be responsible for the SRT; the SRT reading is an average of 50 samples, whereas the other sensors report immediate values at 1-minute intervals.

We capture the diagnostics discrepancies quantitatively besides the visual representation to identify the causes for the clustering divergences—from (6.11) to (6.13). Fig. 6.8 depicts the score of the discrepancies heatmaps among the RBX clusters for each sensor variable. The plots show where the difference lies for each system cluster compared to the others. Fig. 6.8f and 6.8g provide the aggregate summary score $\bar{\psi}_g^m$ and the generated root-cause detection flags R_g^m, respectively. The illustrations indicate that the SCH sensor is the main contributor in most clusters, and most clustering CL-1 sensors exhibit discrepancies. The results validate that the proposed approach successfully captured the summary root causes visually illustrated in the interconnection heatmaps and clustering dendrograms. The summary of root causes is essential for facilitating diagnostics—particularly when the number of clusters or sensors is large and examining extensive heatmaps or dendrograms might be difficult.

Our proposed data-driven interconnection analysis is in conformity with the expected behavior of the RBX-RM systems. The QIE cards are placed nearby, and their environmental characteristics, such as temperature and humidity readings, correlate strongly. The SPV and SPC sensors are from the Peltier system that controls the temperature of the RM internal systems. External humidity controllers such as nitrogen gas regulate the humidity levels of the RMs. The interconnections between the humidity of the SiPM SCH and QIE Q[1-4]H sensors are not strong due to divergence in some of the RBX systems—CL-4 and CL-5; the HCAL external humidity



(a)

(b)

(c)

(d)

(e)

(f)

Figure 6.5: Multivariate sensor interconnection clustering dendrogram per RBX-RM system cluster using sensor clustering threshold $\alpha^s = 0.05$. Plots from a) to e) correspond to the RBX clusters 1 to 5, and f) is an average from all RBX clusters that indicates significant discrepancies in the SRT and SCH sensors, and the interconnection strength.

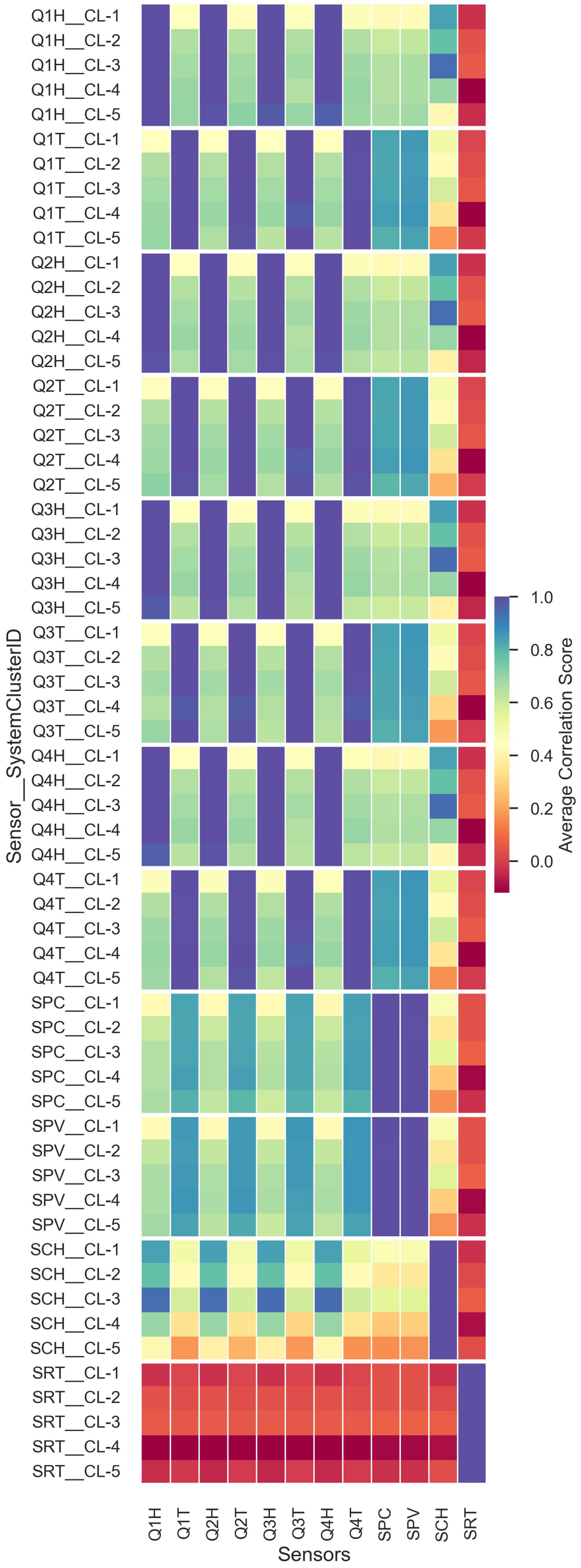
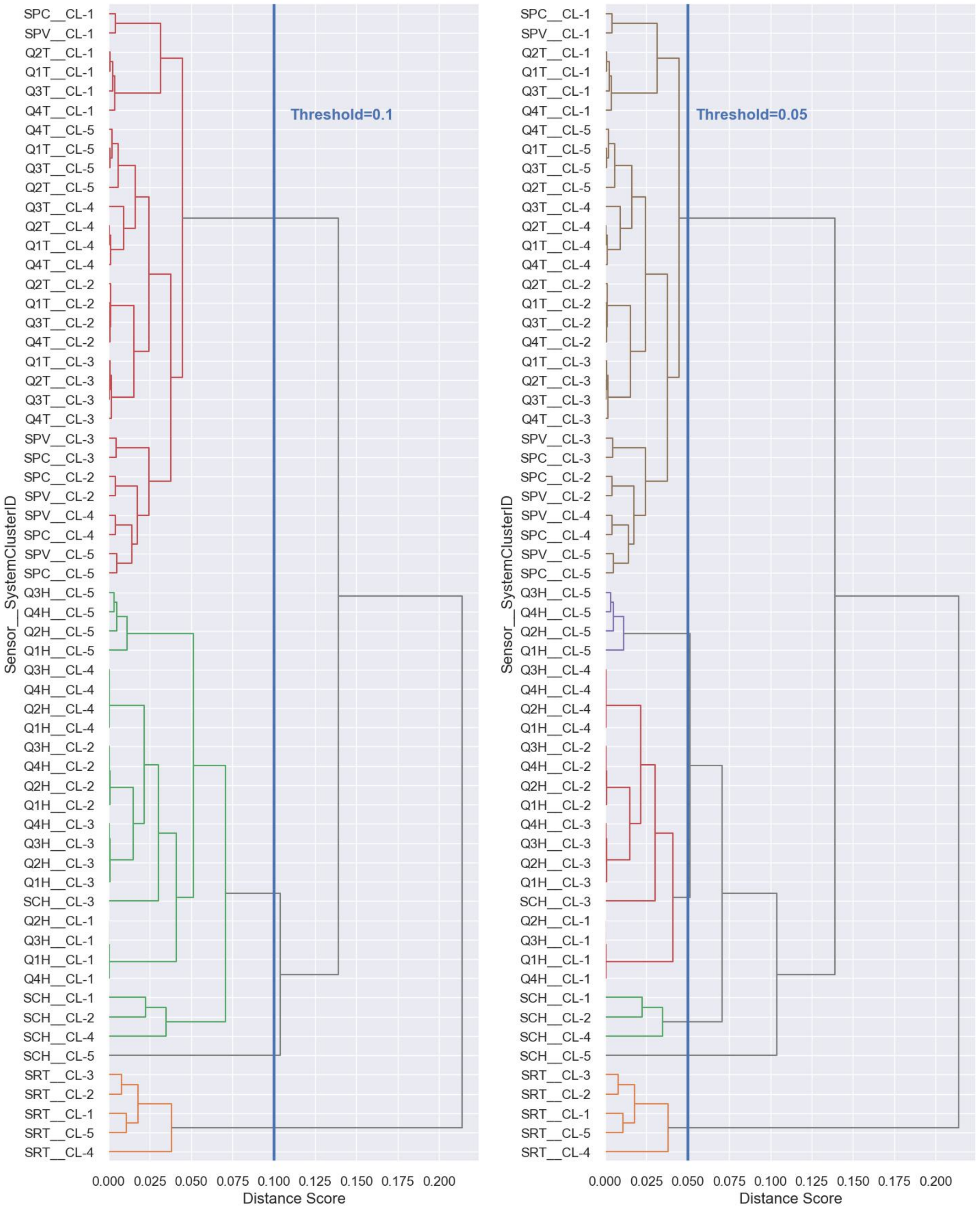


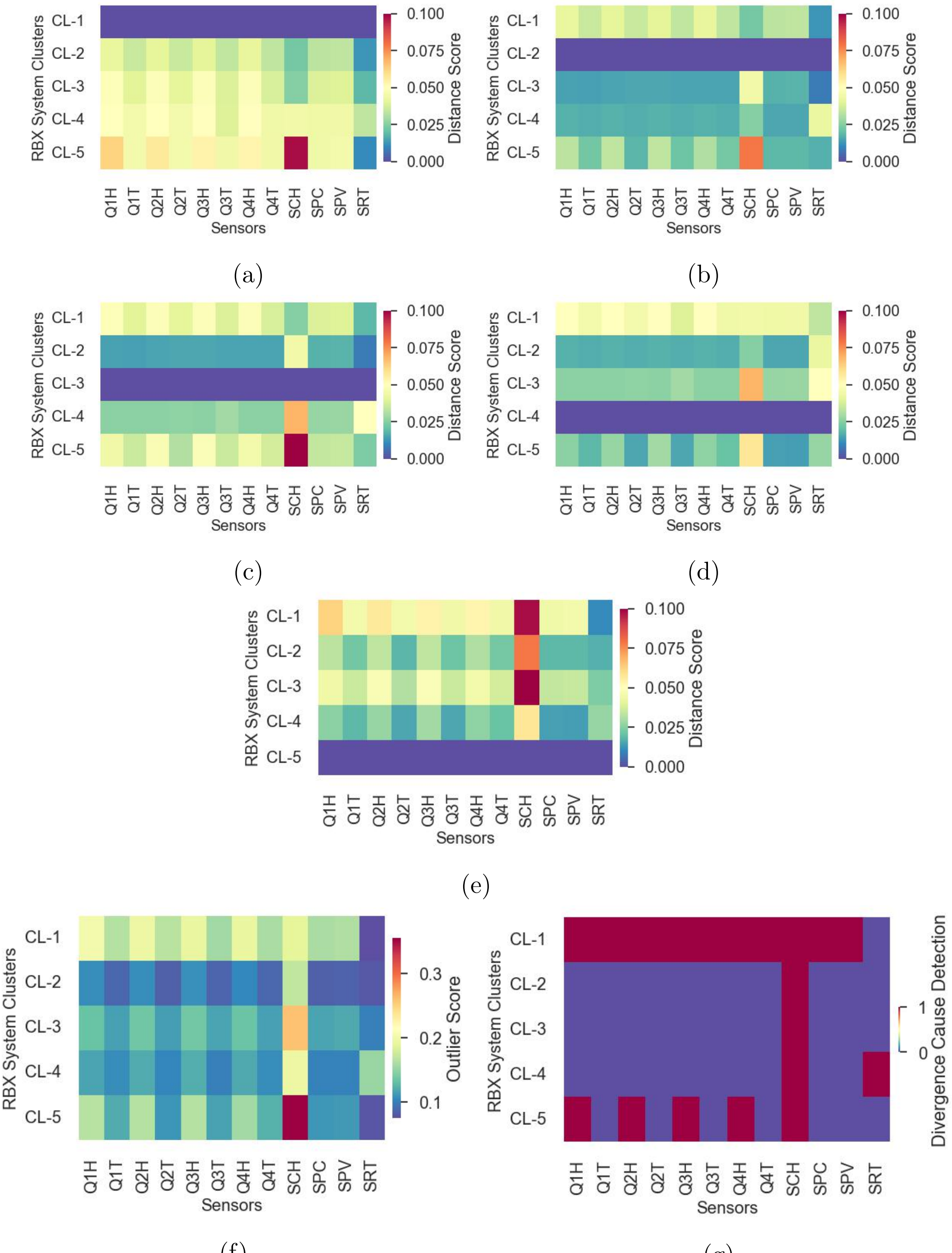
Figure 6.6: The heatmap of the RBX-RM system clusters interconnection on the multivariate sensors. The group boxes show the interconnection strength of the sensors on the x-axis for system clusters on the y-axis. The divergent colors within each box indicate outlier characteristics—e.g., SCH, Q[1-4]H, and Q[1-T] sensors.



(a)

(b)

Figure 6.7: Dengdrogram of sensors interconnections across RBX system clusters: a) at $\alpha^m = 0.1$, and $\alpha^m = 0.05$. Visual representation of how the sensors are clustered together at multi-system cluster levels.



(a)

(b)

(c)

(d)

(e)

(f)

(g)

Figure 6.8: Divergence root-cause detection using the difference in sensor interconnections among the RBX-RM systems clusters. Plots from a) to e) illustrate the sensor divergence score ψ_g^m of each RBX cluster 1 to 5, respectively, and color bars show the strength of discrepancy. The plots in f) and g) are the aggregate divergence scores $\bar{\psi}_g^m$ and the root-cause detection after threshold $\alpha^\phi = 0.15$, respectively—indicating the noticeable divergence in the SCH sensors in all clusters, Q[1-4]H for CL-1 and CL-5, and SRT for CL-4.

controllers regulate a group of RBXes that may cause divergence in humidity measurement among the RBXes. Fig. 6.9 and 6.10 displays the time series data of the sensors grouped by RBX system clusters—validating the accuracy of the generated knowledge of the interconnection analysis that similar RBXes are grouped, diverging patterns in the SCH sensors across clusters in October and November, and humps in the Q[1-4]H sensors of CL-1 at the beginning of September.

6.1.4 Summary

We have developed a simple online mechanism to discover the interconnection of systems and sensors—enabling fast detection of divergent behaviors among multivariate environments. Our experiment has demonstrated the promising performance of the proposed approach in detecting outlying behaviors on the multivariate sensor data of the readout systems of the Hadron Calorimeter. The results have established the approaches' capability to effectively cluster expected system behavior while identifying diverging characteristics. The method enables fast and computationally efficient discrepancy discovery through multi-level clustering analysis along with several heatmaps and clustering illustrations.

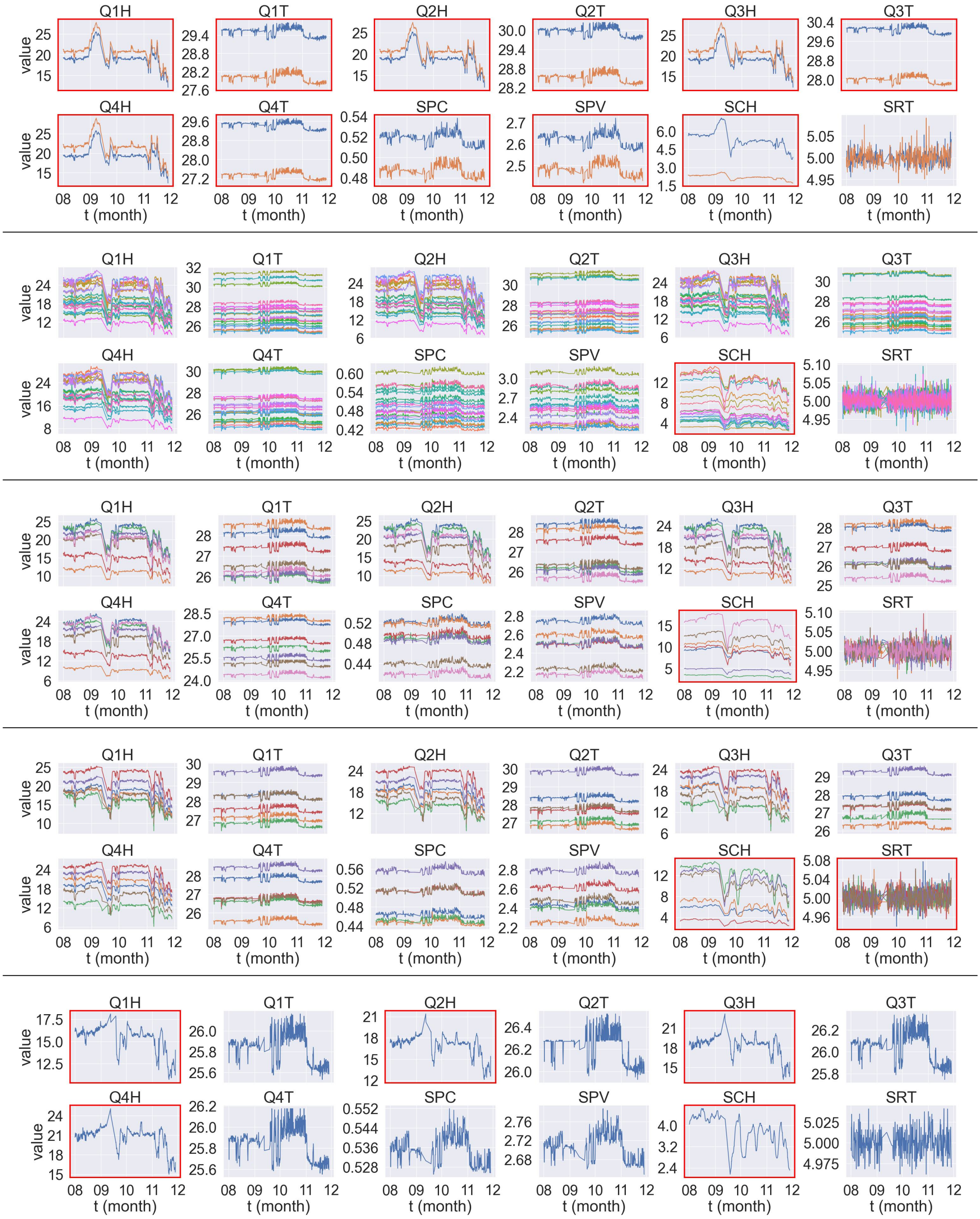


Figure 6.9: Sensor data of the RBX clusters 1, . . . , 5 (top to bottom). The line colors belong to the member RBXes per cluster. Diverging patterns in the SCH across the clusters in October and November; bigger humps on the Q[1-4]H at the beginning of September and smaller jumps on the Q[1-4]T, SPV, and SPC in cluster-1 at the end of September. The root-cause sensors—contributed most to the system clustering divergence—are highlighted with a red box.

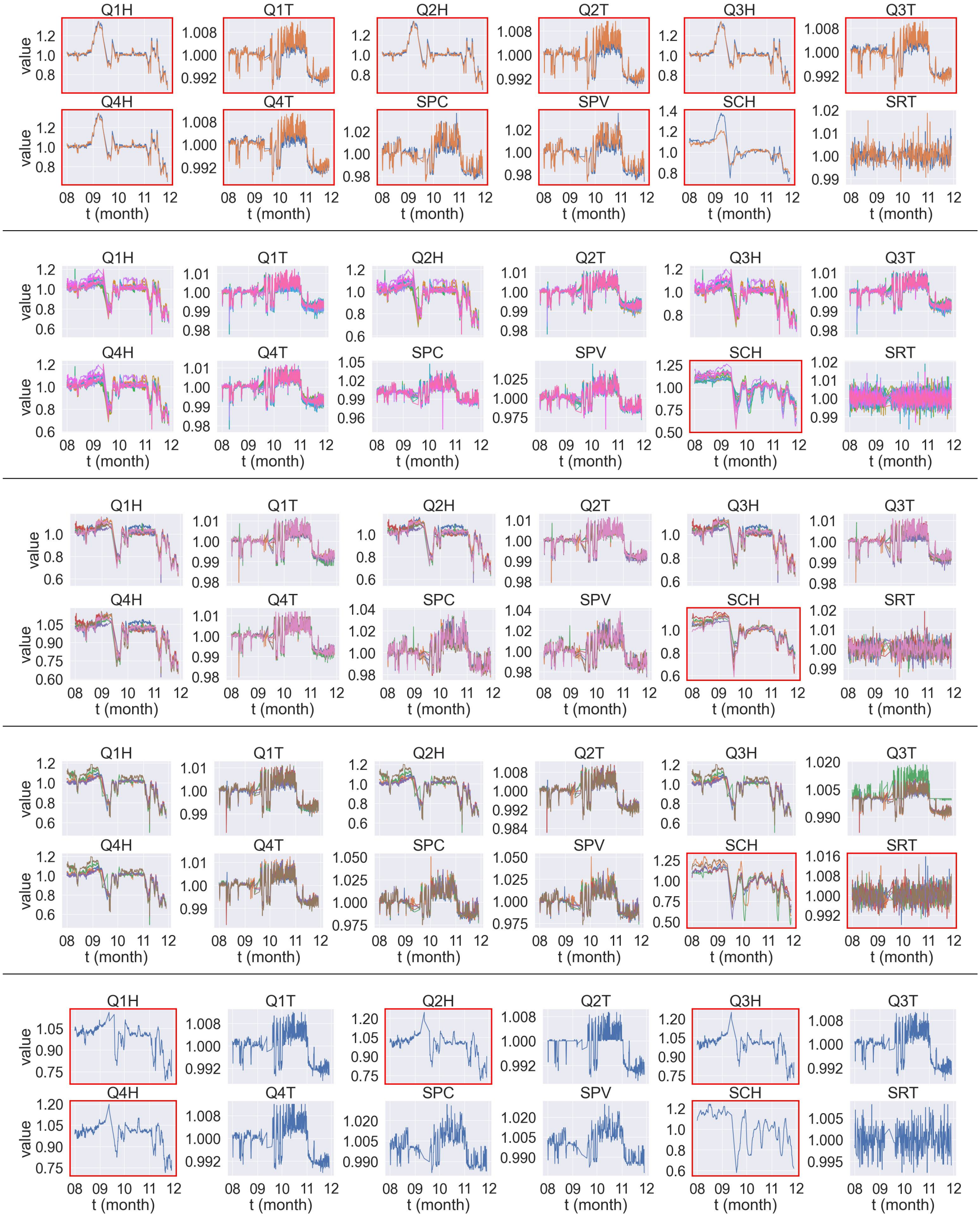


Figure 6.10: Normalized sensor data—divided by median value—of the RBX clusters $1, \ldots, 5$ (top to bottom) for enhanced illustration. The line colors belong to the member RBXes per cluster. Diverging patterns in the SCH across the clusters in October and November; bigger humps on the Q[1-4]H at the beginning of September and smaller jumps on the Q[1-4]T, SPV, and SPC in cluster-1 at the end of September. The root-cause sensors—contributed most to the system clustering divergence—are highlighted with a red box.

6.2 Scalable Temporal Anomaly Causal Discovery in Large Systems

Modern large systems often generate large amounts of binary alarm flags from their monitoring systems. Extracting anomaly causality facilitates diagnostics once a system fault is detected. Learning causal graphs imposes a significant computational burden that restrains the applicability of most existing methods in realtime and large-scale deployments. The distinct characteristics of binary anomaly data—the meaning of state transition and data sparsity—challenge existing causality learning mechanisms. We develop an unsupervised causal graph network approach of causal discovery (AnomalyCD) on binary anomaly data through a computationally efficient mechanism. The AnomalyCD framework consists of multiple strategies to overcome the challenges of generating anomaly causality graphs through unsupervised online anomaly detection, sparse data and link compression, and edge adjustment approaches. We validate the performance of this framework on three datasets: monitoring sensor data of the readout-box system of the HE, a public data set of information technology monitoring, and simulated causal binary anomaly data. The result demonstrates the significant reduction of the computation overhead and moderate enhancement of the accuracy of causal discovery on time series binary anomaly data.

Anomaly detection algorithms are commonly employed in industrial monitoring systems to capture anomalies that require attention—improving efficiency, safety, and reliability while lowering maintenance expenses [213, 217]. Machine learning for anomaly detection (AD) has been presented for front-end diagnostics sensors [41,70], and DQM of the CMS experiment [40,67]. Discovering causality from a broader range of sensors—e.g., the detector backend system components—for captured anomalies is essential to facilitate fault diagnostics through root cause analysis. Causal knowledge of direct and indirect effects, interaction pathways, and time-lags can help understand fault root causes and model physical systems to predict the effect of anomaly occurrence or interventions [228].

The propagating nature of malfunctions makes fault diagnosis challenging in most multivariate processes [213]. Causality knowledge of faults is traditionally acquired through inductive and deductive risk analysis using variants of failure mode and effects analysis and fault tree analysis, respectively [203]. These approaches provide rules for modeling expert knowledge for prior known malfunctions; they may also incorporate querying mechanisms [364]. The approaches require extensive domain knowledge from many experts and are a time-consuming process besides the ambiguity and incompleteness in large systems [203]. Data-driven approaches directly learn causality from the data collected by the sensor monitoring systems [199,200,209,211,213,218,219,365]. The data-driven method can further be categorized as a supervised and unsupervised approach: 1) the supervised RCA methods usually treat the RCA as a fault localization classification problem; an RCA model is trained on labeled cause-effect data—the effect attributes are used as features to predict classes of known root causes [199,200,364], and 2) unsupervised learning RCA is considered a parsimonious and flexible—incorporates previously unknown faults—approach [219] since data annotation is expensive, and not all faults can be experimentally injected [200]. Recent approaches have proposed integration causal discovery (CD) within the AD models [218, 219]. AD and CD often remain highly intractable due to widely diverse operational modes, disparate data types, and complex fault propagation mechanisms [199, 200, 217]. Recent advances in AD for large complex systems in Industry 4.0 focus on specific subsystems because of the curse of dimensionality, data annotation challenges, and the need for accuracy improvement—resulting in multiple AD models for monitoring different subsystems. Identifying the causes of an anomaly thus involves investigating a more extensive set of monitoring variables across subsystems that the trained AD models do not cover. An end-to-end framework for anomaly CD that addresses the challenges pertaining to large systems is of interest in various domains.

We propose a causal discovery framework (AnomalyCD) consisting of several methods to address various challenges related to generating causality knowledge discovery on sparse multivariate anomaly TS data. We employ anomaly alert aggregation from multiple systems ameliorated with a temporal online AD method, sparse data handling, anomaly causal graph structure learning, and Bayesian network modeling for causality inference. We focus on discovering anomaly causality from binary flags because flag data lessens the impact of heterogeneity of the outputs of diverse AD models, and provides data normalization. Although there are some recent efforts for inferring empirical causal graphs from binary data [206] or outlier signals [207], there is still a gap in addressing the unique challenges of TS binary anomaly data for large systems; TS binary anomaly data has distinct characteristics—different from ordinary categorical data—having transition awareness from flag 0 to flag 1—anomaly occurrence—and being exposed to severe data sparsity because of long continuous uniform regions without status changes. The existing causality learning mechanisms do not effectively handle these characteristics. We present a conditional independence test that incorporates anomaly data characteristics together with a constraint-based PCMCI algorithm—a popular constraint-based estimator for CD in time series [20, 86, 236, 237])—to generate the anomaly causal DAG structure and BN [241] to query causality inference. MicroCause [202] employ PCMCI [20, 86] similar to our work, but their effort in addressing binary anomaly data and computational cost—one of the major bottlenecks in TS causal graph discovery—is very limited [202]. The computational cost proliferates when the number of variables and sample size increases and limits the applicability of existing approaches for large-scale systems. We propose a simple but promising compression method for binary anomaly-flag data that substantially reduces the computational cost of the causal graph learning process.

We have applied the proposed framework for monitoring of multivariate sensor data of the HE RBX modules [59]. The results show that our approach accurately detects outlier behaviors and generates causal networks consistent with the actual physical circuit connections and environmental associations. The proposed framework substantially reduces the computational cost of causal graph learning—making it more efficient.

6.2.1 Dataset Description

The HCAL use case dataset includes sensor readings from the four RMs of all thirty-six RBXes in the HCAL Endcap detector. Each RM has twelve diagnostic sensors, four from the SiPM control card system and eight from four QIE card systems (see Table 6.2). The dataset was obtained from the HCAL software monitoring system (ngCCM server) from August to November 2022. The monitoring sensor data comprises four-month data of around 20.7M samples (around 12K per sensor for a given RM). The dataset contains irregularly sampled and considerably sparse data—a few around-minute interval samples are logged every eight and two hours for the SiPM control card and QIE card sensors, respectively. We utilized data from all four RMs of the RBX-HEP07 at one-minute intervals for the online AD and CD evaluation; the RBX-HEP07 is one of the RBX with diverging behavior from our interconnection analysis study in Section 6.1. We removed the extended reading gaps due to various non-physics activities on the LHC and interpolated—up to eight-hour gaps—the remaining time regions into one-minute intervals in data preprocessing. The final dataset contains 100K samples per RM per sensor—4.8M data samples in total.

Table 6.2: Data variables description.

No.	**Notation**	**Variable Name**	**Remark**
1	SPV	PELTIER_VOLTAGE_F	Voltage of the SiPM Peltier temperature controller
2	SPC	PELTIER_CURRENT_F	Current of the SiPM Peltier temperature controller
3	SRT	RTDTEMPERATURE_F	SiPM CC temperature averaged over 50 samples
4	SCH	HUMIDITY_F	SiPM CC humidity
5	Q1H	1-B-SHT_RH_F	QIECARD 1 humidity
6	Q2H	2-B-SHT_RH_F	QIECARD 2 humidity
7	Q3H	3-B-SHT_RH_F	QIECARD 3 humidity
8	Q4H	4-B-SHT_RH_F	QIECARD 4 humidity
9	Q1T	1-B-SHT_TEMP_F	QIECARD 1 temperature
10	Q2T	2-B-SHT_TEMP_F	QIECARD 2 temperature
11	Q3T	3-B-SHT_TEMP_F	QIECARD 3 temperature
12	Q4T	4-B-SHT_TEMP_F	QIECARD 4 temperature

6.2.2 Methodology

We present a framework for anomaly causality diagnostics that addresses the CD of binary anomaly data challenges. The proposed framework comprises two main modules—causal graph discovery and causality inference (see Fig. 6.11). The causal graph discovery prepares temporal causal networks and trains a Bayesian inference model on input binary anomaly-flag data streamed from previously trained (or online) AD systems. The causality inference modules handle user queries with observation conditions and provide causality and conditional probabilities using the

Bayesian model. The causality inference queries on the causal graph network include:

- *Conditional probability inference:* Probabilistic inference allows users to query the BN model for any anomaly occurrence marginal distribution. The inference module provides features to estimate the causal effect between two variables on a given observed anomaly evidence. We employ *variable elimination* algorithm—one of the most widely used exact inference techniques for solving Bayes' equations—to estimate conditional distributions over a subset of variables from the probabilistic graphical BN models.

  $$C_i = \wp(\mathcal{B}, x_i, S) \tag{6.14}$$

  where C_i is the conditional probability of the i^{th} sensor for its anomaly flag state $s_i = 1$ given the observed evidence of states of $S = \{s_j \in \{0, 1\},\ \text{for}\ \forall j \neq i\}$. The $\wp$ is the inference engine with the BN $\mathcal{B}$.

- *Check for causality*: Infers the status of common cause between two sensors with active anomaly flag given evidence of anomaly condition status of the other sensors. It checks if there is an active trail, or d-connection between the start and end nodes, given that the evidence is observed.

- *Infer root-causes networks:* Generates the ancestral DAG graph of the sensor nodes, which have a causal effect on the active anomaly flag on the given sensors.

- *Infer influenced networks:* Executes the do-intervention operation on the BN model where an active anomaly flag is set on given causal sensors to generate the influenced or affected DAG network. The do operation removes all incoming edges to variables in nodes and marginalizes their CPDs only to contain the variable itself.

We have previously trained deep-learning AD models on FE electronics of the HE detector—including the ngCCM and RM and data quality monitoring (DQM) [41, 77, 78]. The HE-ngCCM AD model monitors the *clock and control module* of the 36 RBX—each with 28 sensors [41]. The HE-RM AD model, which adopted Ref. [41], monitors the *readout module* of the RBXes (each RBX has 4 RMs, and each RM has 113 sensors). The DQM-AD monitors 3D spatial data of around 7K physics *acquisition channels*—48 per RM. The HCAL also comprises several other backend (BE) electronic components in addition to the FE circuits; it is often essential to investigate several systems in the pipeline—both the FE and BE electronics of the HCAL—to diagnose faults and identify root causes. We incorporate online temporal AD to enable detecting anomalies on variables that are not actively monitored by trained models—e.g., selected sensors from the FE and BE. Since we have already discussed each of the above FE AD models in the previous Chapter 4 [41, 70, 77, 78], we will focus below on the new addition—the proposed online AD and CD approaches.

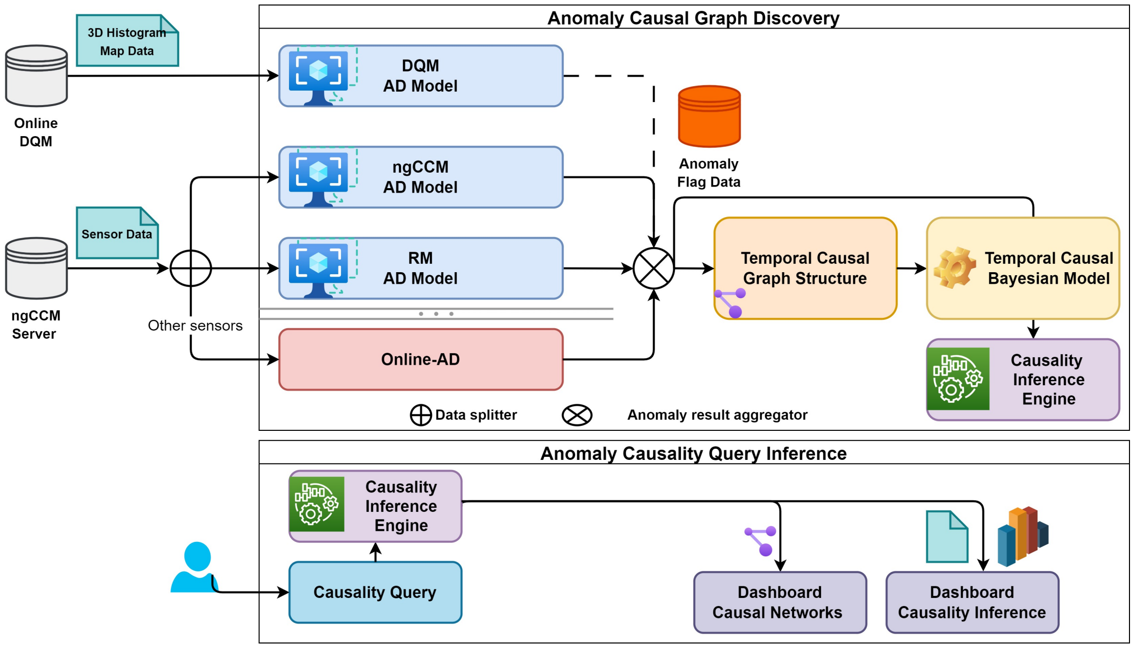

Anomaly Causal Graph Discovery
3D Histogram Map Data
Online DQM
DQM AD Model
ngCCM AD Model
Anomaly Flag Data
Sensor Data
ngCCM Server
RM AD Model
Other sensors
Online-AD
Temporal Causal Graph Structure
Temporal Causal Bayesian Model
Causality Inference Engine
Data splitter
Anomaly result aggregator
Anomaly Causality Query Inference
Causality Inference Engine
Causality Query
Dashboard Causal Networks
Dashboard Causality Inference

Figure 6.11: Anomaly causal discovery and analysis framework for the Hadron Calorimeter. The approach generates a causal interaction graph network and an inference Bayesian model on binary anomaly-flag data generated from several subsystems with online and prior trained AD models.

Online Time Series Anomaly Detection: We present an online AD algorithm to detect unusual temporal patterns and generate anomaly flags for the sensor variables that trained AD models do not actively monitor; we incorporate the algorithm into the proposed CD framework. Different types of variations can occur in time series data—including long-term trends, seasonal changes, periodic fluctuations, and nonrandom sources of variations [301]; these variations can impact the modeling approach and algorithm choices. The proposed online AD system detects anomalies, such as temporal outliers, slow trend drifts, and spectral outliers, using an ensemble of temporal outlier detection algorithms. Building a generic one-fits-all approach is challenging as the requirements depend on signal characteristics and target application. We propose online AD on univariate TS data to capture some of the typical points and collective anomalies—including transient changes in time and frequency, and gradual signal trend drifts. The approach consists of three TS time- and frequency-domain outlier detection algorithms (see Algorithm 3): 1) detecting temporal outliers, 2) detecting changes in temporal trends, and 3) detecting outliers in the spectral domain.

1. *Temporal outlier detection* (see TemporalOutlierDetection in Algorithm 3): The temporal outlier detection employs MovingSDOutlierDetection)—sliding z-score—and TrendDriftDetection on decomposed time series data. We apply *seasonal and trend decomposition using LOESS* (STL) algorithm [366] to

estimate the trend and residual signals, represented by x_ι and x_ϵ, respectively:

$$\begin{aligned} x(t) &= x_\iota(t) + x_\zeta(t) + x_\epsilon(t) \\ x_\iota(t) &= (h_p * x)(t), \text{ where } h_p(t) = \frac{1}{p}[1, 1, ...1] \end{aligned} \tag{6.15}$$

The additive trend x_ι is obtained by applying a convolution ($*$) filter h_p with the shape of $1 \times p$ (moving average with window length or period p) to the data. The average of this de-trended series (after trend removal) for each period is the returned seasonal component x_ζ, and the final remaining component of the series becomes the residual error x_ϵ.

The value of p can be estimated using period estimation methods such as *auto-correlation function* (ACF), *fast Fourier transform* (FFT), *periodogram* (based on FFT), *summary statistics subsequence* (SuSS), and hybrid of ACE and FFT for oscillatory signals [367, 368]. While the approaches show strong accuracy when dealing with cyclic signals with multiplicative trends, we have observed a decline in their performance for additive trends—particularly those with higher slopes. We employ FFT on $\Delta x(t) : x(t) - x(t-1)$ (in the SignalPeriodEstimation) to enhance the period estimation accuracy in the presence of additive or multiplicative trends. The employed decomposition method assumes an additive trend and single seasonality or cyclic pattern. One may employ multiplicative trend or multi-seasonal component decomposition depending on the expected normal signal characteristics [369].

We utilize a sliding z-score outlier detection algorithm on the detrended residual signal x_ϵ (see MovingSDOutlierDetection in Algorithm 3). The x_ϵ signal is normalized by subtracting the mean μ_w and dividing it by the standard deviation σ_w using a sliding time window (w_θ) to generate the outlier score λ_θ. The sliding window localizes the outlier detection on the signal characteristics at the adjacent time data points. The μ_w and σ_w can be affected by strong outliers in a given data—reducing outlier detection efficacy and gets worse for smaller sliding windows; we utilize thus data quantiles $Q = [10\%, 90\%]$ along with a median centering instead of mean to reduce the sensitivity to outlier contamination. We apply a threshold on the outlier score to generate the outlier flags Λ_θ.

We develop a cumulative-based algorithm to detect trend drifts in the trend signal x_ι (see TrendDriftDetection in Algorithm 3). We estimate the trend score λ_ι using cumulative sum on the first-order difference of the trend x_ι signal, and steps are given in lines 22–32. The trend score resets when more significant jumps are detected—often during system operation configuration changes. We apply threshold α_ι to get the drift outlier flags Λ_ι.

2. *Spectral outlier detection* (see SpectralOutlierDetection in Algorithm 3): We employ spectral residual (SR) saliency detection to identify frequency spectrum or data rate change outliers. The SR method has been employed as

a preprocessing technique for cleaning outliers and transforming data in semi-supervised AD research, as demonstrated in Refs. [41] and [148], respectively. The SR algorithm consists of three major steps for a given univariate sequence data [148]: 1) Fourier transform $\mathfrak{F}$ to get the log amplitude spectrum; 2) calculation of spectral residual; and 3) inverse Fourier transform $\mathfrak{F}^{-1}$ that transforms the sequence back to the time domain and generate saliency outlier scores:

$$\begin{aligned} A(f) &= \text{Amplitude}(\mathfrak{F}(\mathbf{x})) \\ P(f) &= \text{Phase}(\mathfrak{F}(\mathbf{x})) \\ L(f) &= \log(A(f)) \\ L_h(f) &= h_q(f) * L(f) \\ R(f) &= L(f) - L_h(f) \\ \eta(\mathbf{x}) &= \left\| \Im^{-1}(\exp(R(f) + iP(f))) \right\| \end{aligned} \tag{6.16}$$

where $\mathbf{x}$ is the input sequence with shape $n \times 1$; $A(f)$ is the amplitude spectrum; $P(f)$ is the corresponding phase spectrum; $L(f)$ is the log representation of $A(f)$; and $L_h(f)$ is the average spectrum of $L(f)$ which can be approximated by convoluting with $h_q(f)$, where $h_q(f) = \frac{1}{q}[1, 1, ...1]$ is averaging filter with an $1 \times q$ vector. $R(f)$ is the spectral residual—the difference of the log spectrum $L(f)$ and the averaged spectrum $L_h(f)$. The sequence is transferred back to the time domain via $\mathfrak{F}^{-1}$ to get the saliency signal η. We apply a threshold α_η to detect anomaly points using η.

Anomaly Causal Discovery: Anomaly graph CD aims to generate an equivalent DAG representing the causal interaction among the monitored variables. Finding the DAG is intractable in most cases. The upper bound for the size of possible DAGs for order n nodes can be estimated by satisfying Robinson's formula [370] as:

$$\mathcal{N}_{DAG}(n) = \sum_{i=1}^{n} (-1)^{i+1} \begin{pmatrix} n \\ i \end{pmatrix} 2^{i(n-i)} \mathcal{N}_{DAG}(n-i) \tag{6.17}$$

The equation shows the size of classes grows super-exponentially; for example, the $\mathcal{N}_{DAG}$ becomes 25 and 10^{11} for $n = 3$ and $n = 8$, respectively. The existing CD algorithms instead search the partial DAGs (PDAGs) that contain *d-separation* with both undirected and directed edges to identify and represent equivalence classes of a given DAG. The approach relies on equivalence concepts that two DAGs are equivalent if and only if they have the same skeleton—the same nodes and undirected edges: X —Y, and directed edges: $X \to Y$ or $X \leftarrow Y$) and the same set of *V-structure* ($X \to Y \leftarrow Z$. The PDAG of an equivalence class $\mathcal{G}$ is the partial directed graph that has the same skeleton as a graph in $\mathcal{G}$ and has a directed edge $X \to Y$ if and only this arc appears in all graphs in the class; for instance, the number of equivalence classes drops from 25 DAGs into 11 PDAGs for $n = 3$.

Algorithm 3 Online temporal anomaly detection

```
procedure OnlineTemporalAnomalyDetection($\mathbf{x}, \alpha_\theta, \alpha_\iota, \alpha_\eta, w_\theta, p_\iota, k_\iota, q_\eta$)
▷ Ensemble online temporal anomaly detection    ▷ $\mathbf{x}$ is a univariate time series signal data
    $\Lambda_\theta,\ \Lambda_\iota \leftarrow$ TemporalOutlierDetection($\mathbf{x}, \alpha_\theta, \alpha_\iota, w_\theta, p_\iota, k_\iota$)
    $\Lambda_\eta \leftarrow$ SpectralOutlierDetection($\mathbf{x}, \alpha_\eta, q_\eta$)
    $\Lambda \leftarrow \Lambda_\theta \cup \Lambda_\iota \cup \Lambda_\eta$
return $\Lambda$
end procedure
procedure TemporalOutlierDetection($\mathbf{x}, \alpha_\theta, \alpha_\iota, w_\theta, p_\iota, k_\iota$)
▷ Temporal outlier detection
▷ $p_\iota$ is the window size of the convolutional filter for trend estimation
    if $isNull(p_\iota)$ then    ▷ for auto period estimation
        $p_\iota \leftarrow SignalPeriodEstimation(\mathbf{x})$    ▷ signal period estimation using FFT
    end if
    $x_\iota, x_\zeta, x_\epsilon \leftarrow TimeSeriesDecomposition(\mathbf{x}, p_\iota)$    ▷ decomposition using STL into a trend, cyclic and residual components
    $\Lambda_\theta \leftarrow$ MovingSDOutlierDetection($x_\epsilon, \alpha_\theta, w_\theta$)
    $\Lambda_\iota \leftarrow$ TrendDriftDetection($x_\iota, \alpha_\iota, p_\iota, k_\iota$)
return $\Lambda_\theta,\ \Lambda_\iota$
end procedure
procedure MovingSDOutlierDetection($x_\epsilon, \alpha_\theta, w_\theta$)
▷ Residual moving standard deviation temporal outlier detection
▷ $\alpha_\theta$ is the detection threshold    ▷ $w_\theta$ is sliding time-window size
    $\lambda_\theta \leftarrow []$
    for $\mathbf{x}_\omega \in GetSlicedTimeWindowData(x_\epsilon, w_\theta, step = 1)$ : do    ▷ get of data slice from sliding time-windows
        $\mu_w,\ \sigma_w \leftarrow GetStats(\mathbf{x}_w)$    ▷ sliding window median and standard deviation using quantile $Q = [10\%, 90\%]$
        $\lambda_\theta \leftarrow Append(\lambda_\theta, |\mathbf{x} - \mu_w|/\sigma_w\ given\ \sigma_w \neq 0)$    ▷ outlier score
    end for
    $\Lambda_\theta \leftarrow \lambda_\theta > \alpha_\theta$    ▷ outlier flag
return $\Lambda_\theta$
end procedure
procedure TrendDriftDetection($x_\iota, \alpha_\iota, k_\iota$)
▷ Cumulative sum-based trend drift outlier detection
▷ $\alpha_\iota$ is the detection threshold
▷ $k_\iota$ is a scaling constant
    $d_\iota \leftarrow \Delta(x_\iota) : x_\iota(t) - x_\iota(t-1)$    ▷ step change of trend data points
    $\mu_{d_\iota} \leftarrow MEDIAN(|d_\iota|)$    ▷ average step change on the trend
    $\bar{d_\iota} \leftarrow |d_\iota| > k_\iota \mu_{d_\iota}$    ▷ check for large step changes in the trend
    $\lambda_\iota \leftarrow Zeros(LENGTH(x_\iota))$
    for $t_r, \bar{d}_{\iota r} \in GetContinuousRegion(\bar{d_\iota})$ : do
        if $\bar{d}_{\iota r}$ then
            $\lambda_\iota(t_r[j]) \leftarrow \sum_{i=t_r(1)}^{t_r[j]} d_\iota(t_r),$ for $j = [1, ..., N_{t_r}]$    ▷ calculates trend drift score using cumulative sum
        end if
    end for
    $\Lambda_\iota \leftarrow \lambda_\iota > \alpha_\iota$    ▷ outlier flag
return $\Lambda_\iota$
end procedure
procedure SpectralOutlierDetection($\mathbf{x}, \alpha_\eta, q_\eta$)
▷ Spectral residual (SR) temporal outlier saliency detection
▷ $\alpha_\eta$ is the detection threshold
▷ $q_\eta$ is siding spectral kernel size
    $\eta \leftarrow SpectralResidualSaliency(\mathbf{x}, q_\eta)$    ▷ spectral residual saliency score
    $\lambda_\eta \leftarrow \frac{\eta - \overline{\eta}}{\overline{\eta}}$    ▷ normalized outlier score
    $\Lambda_\eta \leftarrow \lambda_\eta > \alpha_\eta$    ▷ outlier flag
return $\Lambda_\eta$
end procedure
```

The process of computing a DAG for the temporal CD of anomalies in large systems with multiple variables and large data is challenging. We have incorporated several methods to address CD challenges from binary anomaly data; the approaches include data size compression using sparse handling algorithms, sparsity-driven prior time-delay link assumption compression, one-side edge independence testing for anomaly triggering from flag 0 to 1 transitions, and post-processing link adjustment to avoid cyclic edges on different time-lags. The techniques help to alleviate the computational burden and improve the accuracy of the anomaly CD.

Our proposed anomaly CD approach includes three main modules: data pre-processing, causal graph structure generation, and building the Bayesian inference model. The approach infers temporal causal interaction among monitoring variables or sensors using time-lagged and contemporaneous CD algorithms (see Fig. 6.12). We employ PCMCI for its accuracy in temporal CD and propose additional augmentation algorithms to enhance its effectiveness with particular challenges of time series binary anomaly data. The PCMCI may result in cyclic links even if the expected causal graph is acyclic due to errors in estimation when dealing with a long sequence of overlapping binary anomaly regions among sensors [208]. We reduce the computational cost and enhance the accuracy of causal graph building at the data pre-and post-processing stages through sparse data and graph edges compression, CI testing sensitive to binary flag transitions, and link pruning.

- *Sparse data handling*: The computational cost of causal graph discovery methods varies with the numbers of variables N, data sample sizes n, and maximum time-lag $\tau_{\max}$ [236]. The complexity of the conditional independence (CI) test, $X \perp Y | Z$: X independent of Y conditioned on Z, is one major factor affecting the computational workload in a constraint-based algorithm [236]; for instance, partial correlation CI test scales with a complexity of $\mathcal{O}(n(N\tau_{\max})^2)$. Optimizing computational efficiency by reducing n to n' for a given N and $\tau_{\max}$ is advantageous. The processing speed gain also allows longer $\tau_{\max}$ for extended causality search on n' without additional cost. We propose to lessen the computational cost by reducing sample data size—exploiting the anomaly data sparsity. Data sparsity is inevitable in the TS binary anomaly data since anomalies occur rarely and may persist for some time. Our module incorporates sparse data handling that compresses the long-time regions with uniform anomaly status so that anomaly CD can better be captured from the anomaly status transitions (see Algorithm 4). We preserve the first l_m indices of the region to capture the time-lag causality that ensures inference within time-adjacent anomaly occurrences while substantially reducing the size of the sparse data. We compress the time regions into time length l_m slightly greater than the causality searching $\tau_{\max}$ to avoid false adjacency between anomalies at different time stamps after the uniform anomaly regions compression. The l_m also regulates the contribution between collective trend drift anomalies and transient anomalies; longer l_m increases the influence of the collective anomalies on the causality estimation. Our method aims to reduce the sample size to partially resolve the computational burden in the GCM learning process for large-scale deployment.

We build a causal graph network generation method for TS binary anomaly data; we leverage the PCMCI algorithm [20,228] to learn the graph skeleton $\mathcal{G}(\mathcal{V},\mathcal{E})$ from a TS data, where $\mathcal{V}$ is the set of sensor node vertices $\upsilon, \nu \in \mathcal{V}$, and $\mathcal{E}$ is the edge matrix $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} : \varepsilon(\upsilon,\nu) \in \mathcal{E}$. We limit the discussion to our algorithms and leave readers to refer to Refs. [20, 86, 228] for a comprehensive discussion on the working principle of the PCMCI algorithm. Our modification revolves around customization for handling binary anomaly data, reducing computation by prior link assumption, and removing cyclic links from causality confusion in long overlapped anomaly regions.

- Constraint-based graph CD methods rely on CI tests to estimate links among variables. The independence score function $\mathcal{I}(X,Y)$ answers CI queries of the form $X \perp Y \mid Z$ on given dataset $\mathcal{D}$ that the variables assumed to be generated independently from some (unknown) Bayesian system as:

$$\mathcal{I}(X,Y) : P(X,Y \mid Z) = P(X \mid Z)P(Y \mid Z) \tag{6.18}$$

The independence test may result in Type I errors (false positive that rejects true independence) and Type II errors (false negative that accepts false independence), as P (derived from $\mathcal{D}$) is an approximate description of the actual underlying system behavior. The trade-off controlling the error is obtained by a significance level threshold α given independence measuring function $f_I(\mathcal{D})$. The p_v, probability of observing independence of the test, is given as:

$$p_v(\alpha) = f_I(\mathcal{D}) > \alpha \tag{6.19}$$

We aim to capture the causality linkages behind binary anomaly data, particularly the transition from being healthy with flag 0 to experiencing an anomaly occurrence with flag 1. The popular CI tests for categorical data, such as statistical *G-squared* [371] and information-theory *conditional mutual information* (CMI) tests [228], may not easily distinguish the significance of the anomaly transition behavior; these methods could result in incorrect causality inferred from the association influenced by the zeros instead of the ones. We propose *anomaly aware CI testing* (ANAC) using a partial-correlation CI test that only considers links with positive associations corresponding to anomaly occurrence. The PCMCI estimates GCM of a given TS variable (anomaly flag Λ_i in our case) with a time-lagged function of the multivariate:

$$\Lambda_i = \mathcal{F}(w_j(t-s), \Lambda_j(t-s)), \text{ for } s = 0, \ldots, \tau_{\max} \text{ and } j \in \mathbf{PA}_i \tag{6.20}$$

where the Λ is the multivariate time series anomaly data; the w is the causal time-lagged strength weights; the $\tau_{\max}$ is the maximum time-lag for causal inference; the $\mathbf{PA}_i$ is the set of the parent nodes or variables of i; the $\mathcal{F}$ is a binary GCM function. The CI test influences the w: the positive w indicates a correlation in increasing—anomaly occurrence—on the prediction Λ_i. We leverage the PCMCI [20] with ANAC to detect anomaly occurrence causality by selecting causal time-lags with $w > 0$. The partial correlation is estimated through *linear ordinary least squares (OLS) regression* and positive *Pearson's correlation* (ρ) CI test on the residuals. To test $X \perp Y \mid Z$, first Z is regressed out from X and Y using the OLS regression model:

$$\begin{aligned} X &= \beta_X Z + \epsilon_X \\ Y &= \beta_Y Z + \epsilon_Y \end{aligned} \tag{6.21}$$

The independence of the residuals is evaluated using Student's t-test on the $\rho\,(\epsilon_X, \epsilon_Y)$ to generate the p_v.

The complexity of CD remains for high dimensional data with large N despite the partial reduction of the computation by the sparse data compression. We propose a prior link assumption method to reduce the link searching space. The proposed link assumption approaches aim to relieve the computation by reducing the number of conditional tests and enhancing accuracy in capturing causality among the variables. We integrate a prior link assumption that excludes self-lag causality and non-overlapping links on the binary anomaly data that $\mathbf{PA}_i \in X_j(t-s)$, where $j \neq i$. The assumption of self-lag exclusion reduces the link search space computational cost from $\mathcal{O}(n'(N\tau_{\max})^2)$ to $\mathcal{O}(n'N(N-1)\tau_{\max}^2)$. We present the sparse link handling algorithm to exclude links from temporally non-overlapping signals (see Algorithm 5). The algorithm extends the anomaly flag regions by $\tau_{\max}$ to measure the time-lagged co-occurrence among a pair of variables; a link is not considered during the learning phase if its overlap score is below a certain threshold—e.g., zero indicates no overlap.

- Multiple sensor variables can report anomaly flags simultaneously for continuous time ranges that might cause the PCMCI [20, 228] to generate PDAG, which includes spurious edges—multiple time-lags, undirected, bidirected, or cyclic links—when dealing with a temporal anomaly data. Bidirected edges also occur when there is no time delay causality; the correlation-based CI tests are symmetric and cannot distinguish the edge direction at $t = 0$. We present a pruning algorithm as post-processing to overcome this challenge (see Algorithm 6). The algorithm groups linked nodes and keeps the lag with the highest weight or earliest time-lag from the reported causal lags; higher link weights indicate strength and older time lag corresponds to earlier causality—temporally close to the transitioning edges. We employ a different CI test (the chi-square test) to direct the bidirected edges at $t = 0$ when the correlation CI test falls short in detecting the directions. The positive transition association of the anomaly flags on the directed edges holds even in the chi-square test since the edges are first detected using positive correlation scores. The pruning excludes the spurious links caused by the continuously overlapped regions and enables the correlation test to generate a curated DAG that meets the requirement for building the inference BN model for causality query in the later stages—BN requires a GCM without cyclic links.

Building Bayesian Network Model: We employ a BN to qualify and enable causality inference for user query conditions beyond having a static graph skeleton. We utilize a *Bayesian parameter estimator* [229] using a *Bayesian Dirichlet equivalent uniform prior scoring* to learn the Bayesian CPD parameters of the discovered network skeleton structure of the PCMCI temporal GCM. The edges of the generated temporal causal graph $\mathcal{G}((\upsilon, \nu), \varepsilon(w, s))$ have weight w and time-lag s attributes. To build a temporal causal BN, we reformulate $\mathcal{G}(\mathcal{V}, \mathcal{E}) : (\upsilon, \nu) \rightarrow (\upsilon_s, \nu)$, $\varepsilon(w, s) \rightarrow \varepsilon(w_s)$ that source node υ with an edge $\varepsilon(w, s)$ is represented by new nodes υ_s and an edge $\varepsilon(w_s)$ for every active time-lag s. We prepare the data for the υ_s by unrolling the time series data of the $\upsilon(t)$ into structural data—a new column with υ_s is added by shifting the data ahead by the amount of time-lag s (see Algorithm 7). There are also other tools to build TS BN, such as dynamic BNs, but most are restricted to 2-time step temporal BN (2-TBN) that requires only unit time-lag links or existence of self-lag connections [372, 373].

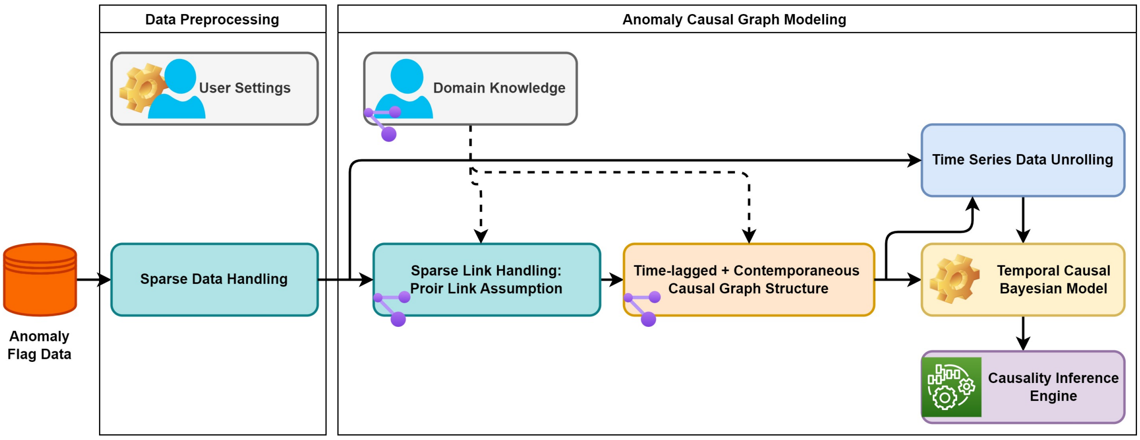

Data Preprocessing
Anomaly Causal Graph Modeling
User Settings
Domain Knowledge
Time Series Data Unrolling
Anomaly Flag Data
Sparse Data Handling
Sparse Link Handling: Proir Link Assumption
Time-lagged + Contemporaneous Causal Graph Structure
Temporal Causal Bayesian Model
Causality Inference Engine

Figure 6.12: Temporal anomaly causal discovery approach diagram. The approach infers causal interaction among monitoring sensor variables from a time series binary anomaly data.

6.2.3 Experiment Results on the HCAL Anomaly Data

We discuss the performance of the proposed time-series anomaly CD and analysis framework using a use cause system RBX-HEP07—one of the RBXes that exhibit divergent behavior in section 6.1.2. We employ the anomaly flags—generated from our proposed ensemble online anomaly detection algorithms—for the CD experiment.

The dataset comprises 12 sensors of each HE-RM system. Several extended reading gaps exist due to various non-physics activities on the LHC. We generated a reading mask—shown in Fig. 6.13—to filter out the irrelevant operational time

Algorithm 4 Data compression on sparse binary anomaly flag

```
procedure SparseBinaryDataHandler($\Lambda, l_m$)
▷ $\Lambda \in \mathbb{R}^{N \times T}$ is a matrix of anomaly flags with $N$ sensors and $T$ time length
▷ $l_m$ is the maximum time length for uniform states data compression
    $\Lambda_c \leftarrow []$        ▷ the compressed anomaly flags data $\Lambda_c \in \mathbb{R}^{N \times T_c}$
    $S \leftarrow AggregateAnomalyState(\Lambda)$        ▷ returns sequence of binary concatenation of from all variables
    for $I, l_s \in GetUniformStateTimeRegions(S)$ : do
▷ $I$ holds indices of the uniform status region with a time length of $l_s$
        $\Lambda_I \leftarrow \Lambda[I]$        ▷ $\Lambda_I$ is the selected uniform status flag region
        if $l_s > l_m$ then
            $I_d \leftarrow GetRangeToRemove(\Lambda_I, l_m)$        ▷ the tail indices of the region, excluding the first $l_m$ data points
            $\Lambda_I \leftarrow CompressBinayData(\Lambda_I, I_d)$        ▷ compressing time length of $\Lambda_I$ by removing $I_d$ data indices
        end if
        $\Lambda_c \leftarrow Append(\Lambda_c, \Lambda_I)$
    end for
return $\Lambda_c$
end procedure
```

Algorithm 5 Sparse link handling with refined prior link assumption

```
procedure SparseLinkHandler($\Lambda, \mathcal{E}, \tau_{\max}, \alpha_\tau$)
▷ $\Lambda$ is multivariate time series binary anomaly data
▷ $\mathcal{E}$ is a prior assumption of directed time-lagged edge links
▷ $\tau_{\max}$ is the causality search maximum time-lag
▷ $\alpha_\tau$ is time-lagged anomaly flag overlap strength threshold
    $d\Lambda \leftarrow \Delta\Lambda : \Lambda(t) - \Lambda(t-1)$
    $S_{d\Lambda} \leftarrow TimeExtendAnomalyRegion(size = \tau_{\max})$        ▷ uses $SlidingWindowSum$ with $window = \tau_{\max}$ and $step = 1$
    $\Lambda_\tau \leftarrow S_{d\Lambda} > 0$        ▷ convert to binary
    for $\Lambda_\tau(i), \Lambda_\tau(j) \in GetUniquePairVariables(\Lambda_\tau)$ : do
        $\lambda_\tau \leftarrow SimultaneousAnomalyFlagCount(\Lambda_\tau(i), \Lambda_\tau(j))$
        if $\lambda_\tau > 0$ then        ▷ check if overlap exists
            $n^i \leftarrow AnomalyFlagCount(\Lambda_\tau(i))$
            $n^j \leftarrow AnomalyFlagCount(\Lambda_\tau(j))$
            $\lambda_\tau^{ij} \leftarrow \frac{\lambda_\tau}{n^i}$        ▷ normalized overlap score from $i$ to $j$
            $\lambda_\tau^{ji} \leftarrow \frac{\lambda_\tau}{n^j}$        ▷ normalized overlap score from $j$ to $i$
            if $\lambda_\tau^{ij} < \alpha_\tau$ then
                $\mathcal{E} \leftarrow RemoveEdges(\mathcal{E}, i, j)$        ▷ remove all edges from node $i$ to $j$
            else if $\lambda_\tau^{ji} < \alpha_\tau$ then
                $\mathcal{E} \leftarrow RemoveEdges(\mathcal{E}, j, i)$        ▷ remove all edges from node $j$ to $i$
            end if
        else
            $\mathcal{E} \leftarrow RemoveEdges(\mathcal{E}, i, j)$        ▷ remove all edges from node $i$ to $j$
            $\mathcal{E} \leftarrow RemoveEdges(\mathcal{E}, j, i)$        ▷ remove all edges from node $j$ to $i$
        end if
    end for
return $\mathcal{E}$
end procedure
```

Algorithm 6 Pruning and adjusting time-lagged causality edges

```
procedure AdjustTemporalCausalityLinks($\mathcal{E}, mOptions = [0, 1]$)
    ▷ $\mathcal{E} \in \mathbb{R}^{N \times 3}$ is a matrix of weighted directed time-lagged edge links $\varepsilon(\upsilon, \nu, w, t)$
    $\mathbf{G}_\varepsilon \leftarrow EdgeGroupMaxWeightedTimeLag(\mathcal{E})$    ▷ groups edges with same nodes and select highest weighted time-lag
    $\mathbf{R} \leftarrow [\,]$    ▷ placeholder for edges to be removed
    $\mathbf{D} \leftarrow [\,]$    ▷ placeholder for undirected edges
    for $\varepsilon_s, \varepsilon_r \in\ GetBidirectLinkedNodes(\mathbf{G}_\varepsilon)$ : do    ▷ the current edge $\varepsilon_s$ and the reverse edge $\varepsilon_r$
        $\varepsilon_p \leftarrow$ CompareBidirectLinks($\varepsilon_s, \varepsilon_r, m = mOptions[0]$)
        if $\varepsilon_p$ is not Edge then
            $\varepsilon_p \leftarrow$ CompareBidirectLinks($\varepsilon_s, \varepsilon_r, m = mOptions[1]$)
            if $\varepsilon_p$ is not Edge then
                $\mathbf{D} \leftarrow Append(\mathbf{D}, [\varepsilon_s, \varepsilon_r])$    ▷ add the edges $\varepsilon_s$ and $\varepsilon_r$ into the undirected bucket
                $\mathbf{R} \leftarrow Append(\mathbf{R}, [\varepsilon_s, \varepsilon_r])$    ▷ add the edge $\varepsilon_s$ and $\varepsilon_r$ into remove bucket
            else
                $\mathbf{R} \leftarrow Append(\mathbf{R}, \varepsilon_p)$    ▷ add the edge $\varepsilon_p$ into remove bucket
            end if
        else
            $\mathbf{R} \leftarrow Append(\mathbf{R}, \varepsilon_p)$    ▷ add the edge $\varepsilon_p$ into remove bucket
        end if
    end for
    $\bar{\mathcal{E}_R} \leftarrow PruneEdges(\mathbf{G}_\varepsilon, \mathbf{R})$    ▷ remove edges in the remove bucket
    $\bar{\mathcal{E}_D} \leftarrow DirectEdges(\mathbf{G}_\varepsilon, \mathbf{D})$ ▷ get DAG for the undirected edges without affecting the directed edges in $\mathbf{G}_\varepsilon$
    $\bar{\mathcal{E}} \leftarrow Merge(\bar{\mathcal{E}_R}, \bar{\mathcal{E}_D})$    ▷ merge pruned and directed edges
return $\bar{\mathcal{E}}$
end procedure
procedure CompareBidirectLinks($\mathbf{G}_\varepsilon, \varepsilon_s, \varepsilon_r, m$)
    $t_s, w_s \leftarrow GetEdgeAttributes(\varepsilon_s)$    ▷ get edge link time-lag and weight of link $s$
    $t_r, w_r \leftarrow GetEdgeAttributes(\varepsilon_r)$    ▷ get edge link time-lag and weight of the reverse link $r$
    $\varepsilon_p \leftarrow None$    ▷ placeholder for the edge to be pruned
    if $m$ is $0$ then    ▷ use link weight values
        if $w_s > w_r$ then
            $\varepsilon_p \leftarrow \varepsilon_r$
        else if $w_s < w_r$ then
            $\varepsilon_p \leftarrow \varepsilon_s$
        end if
    else if $m$ is $1$ then    ▷ link time-lag values (negative)
        if $t_s < t_r$ then
            $\varepsilon_p \leftarrow \varepsilon_r$
        else if $t_s > t_r$ then
            $\varepsilon_p \leftarrow \varepsilon_s$
        end if
    end if
return $\varepsilon_p$
end procedure
```

Algorithm 7 Temporal Bayesian network model generation

```
procedure BayesianNetworkModelGeneration($X, \mathcal{G}$)
    ▷ $\mathbf{X} \in \mathbb{R}^{N \times T}$ is a multivariate time series data
    ▷ $\mathcal{G}$ is a graph network with $\mathcal{G}(\upsilon, \nu, \varepsilon(w, t))$
    ▷ $\mathcal{E} \in \mathbb{R}^{N \times T}$ is a matrix of anomaly flags with $N$ sensors and $T$ time length
    for $\varepsilon(\upsilon_t, \nu) \in\ GetLinkedNodes(\mathcal{G})$ : do
        $t \leftarrow GetEdgeTimeLag(\upsilon_t)$    ▷ get source node time-lag
        if $t < 0$ then    ▷ checks the time delayed causality
            $\bar{x_\upsilon} \leftarrow GetTimeDelayedData(x_\upsilon, t)$    ▷ shifts backward the a time series $x_\upsilon \in \mathbb{R}^{1 \times T}$ by $t$ time steps
            $X \leftarrow Append(\bar{x_\upsilon})$
        end if
    end for
    $\mathcal{B} \leftarrow FitBayesianNetwork(X, \mathcal{G})$    ▷ build Bayesian network model
return $\mathcal{B}$
end procedure
```

regions of the LHC and ensure any detected anomalies are within the normal operation of the calorimeter. We interpolate—up to eight-hour gaps—the remaining regions into one-minute intervals. We utilize data from all RM sensors of the RBX-HEP07 system (RM-1, RM-2, RM-3, and RM-4) to discuss the performance of the proposed methods (see Fig. 6.14). The data comprise short-living transient and time-persistent anomalies, such as trend drifts. We employ data from multiple RMs to capture the global causality of the HE-RM. We will first discuss the performance of the online AD approach and then proceed to the anomaly CD.

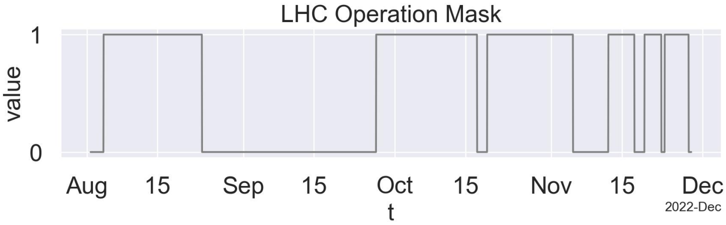


Figure 6.13: The active mask of the LHC operation status from August to December of 2022. The active $mask = 1$ refers to the LHC during its normal physical run operation status—during running collision experiment or idle—whereas the $mask = 0$ corresponds to the LHC under other non-physics operation states—e.g., technical stop and maintenance development.

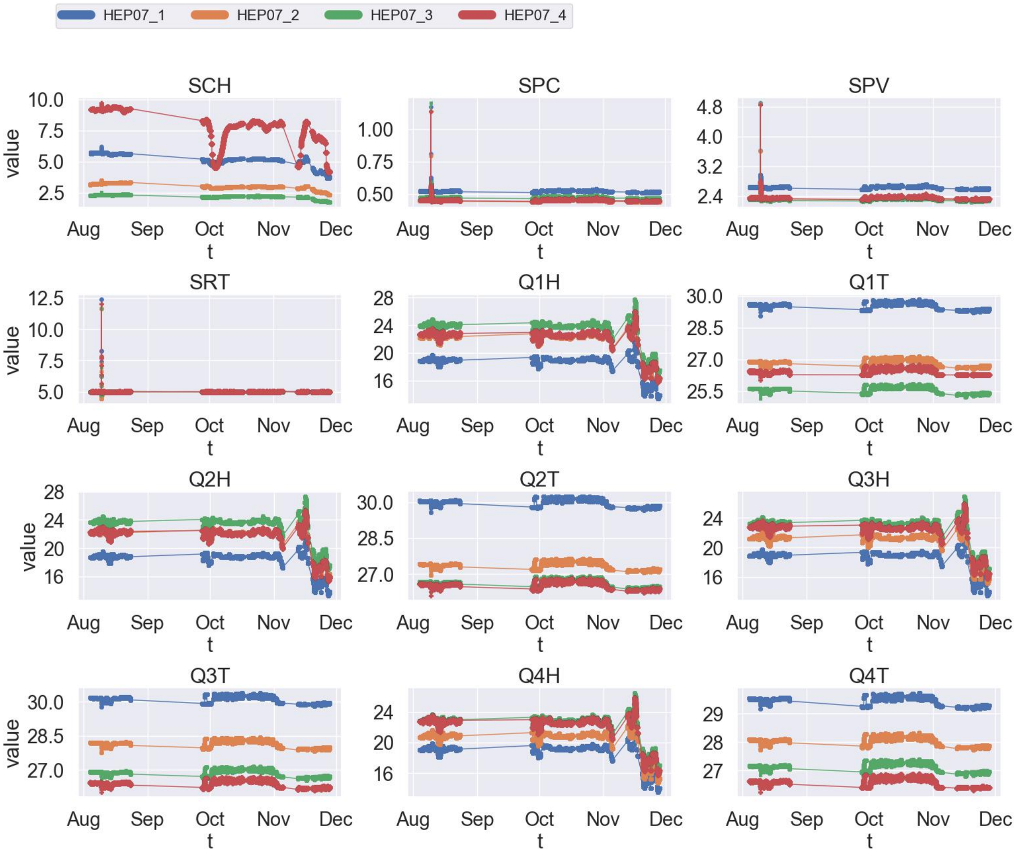


Figure 6.14: Sensor time series reading data from all four RMs of the RBX-HEP07.

Online Anomaly Detection: The proposed ensemble online temporal AD approach involves three methods: 1) global temporal outlier detection, 2) temporal rate drift detection using spectral residual saliency, and 3) trend drift detection (see Section 6.2.2). Table 6.3 provides the hyperparameter settings of the algorithm. We set the anomaly thresholds slightly higher to reduce noise contamination and preserve the causal faithfulness assumption [204, 209, 222].

The LHC has undergone operations that result in distinct signal patterns on the sensors; we have further utilized change point breaks (on 2022-09-27, 2022-10-19, and 2022-11-12) in which the AD system reinitializes. Change point detection algorithms, such as PELT [374] and kernel-based [375], can detect changes in operation automatically from TS signal data; we found their accuracy is sensitive to hyperparameters and unsatisfactory for non-periodic and non-stationary signals—increase false detection.

Table 6.3: Hyperparameter settings for the outlier detection algorithms.

Algorithm	**Settings (at 1 minute sampling interval)**
MovingSDOutlierDetection	$\alpha_\theta = 10,\ w_\theta = 5760$
TrendDriftDetection	$\alpha_\tau = 20,\ p_\tau = 5760,\ k_\tau = 5$
SpectralOutlierDetection	$\alpha_\eta = 35,\ q_\eta = 1440$

Fig. 6.15 depicts the SCH sensors of the four RMs of HEP07 along with the detected transient and trend outliers marked on the outlier score signals of the outlier detection algorithm. The sensors show a drifting trend where they gradually deviate away—dropping or increasing—from their expected optimal values. Fig. 6.16 illustrates all the sensors from RM-1 with marked anomalies. Fig. 6.17 portrays the anomaly flag count from all RMs of the HEP07; the humidity sensors have higher counts due to the detected trend drift anomaly. The time of occurrence of anomalies on the SRT seems temporally correlated with the Q[1-4]T, SPV, and SPC.

We have demonstrated the capability of our proposed online AD approach in detecting several types of outliers on TS data with light computation overhead. The potential limitation of the approach is that it expects adequate healthy data samples; it is challenging to detect outliers without prior knowledge if the outlier is dominant in data—violating the rare anomaly occurrence assumption—and in such cases, using trained AD models is recommended instead. Finding the optimal hyperparameters is also still an open challenge. Data normalization and standardization partially alleviate this challenge, but the hyperparameters may remain dependent on the data or the target outlier characteristics—requires domain knowledge for tuning. Some initial parameter-tuning effort is required when employing the approach in a new environment. Setting adjustment—experimenting with statistical models—is a rather fast process compared to hyperparameter tuning on deep learning models.

Anomaly Causal Discovery: We capture the causal graph from the TS binary anomaly data generated by the online AD algorithm in the previous section (see Fig. 6.18). We set the maximum time-lag $\tau_{\max} = 5$ to search for temporal causality

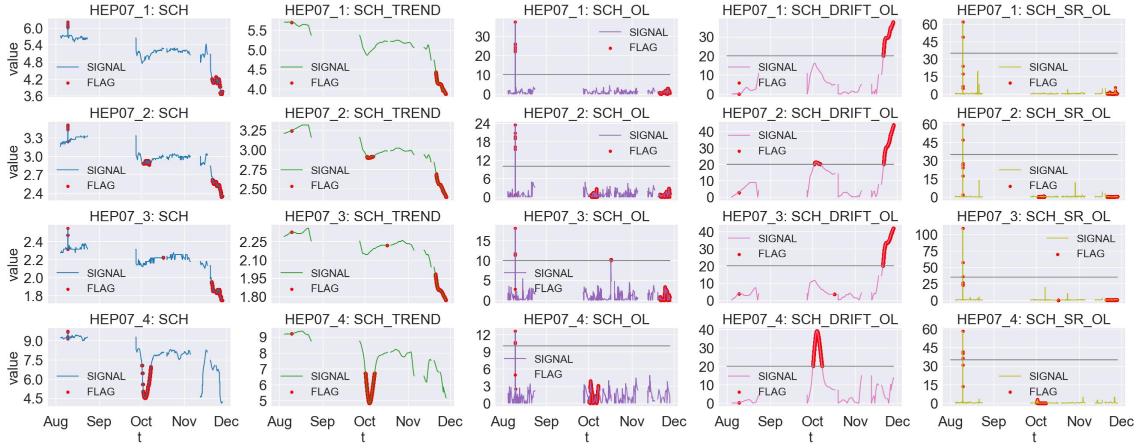


Figure 6.15: Online temporal anomaly detection on RBX-HEP07 SCH sensor. (Left to right) sensor signal, signal trend estimation, Λ_ι, Λ_θ, and Λ_η.

dependency at $t-\tau_{\max}, \ldots, t-1, t$ (equivalent to five minutes) and CI test significance threshold $p_v = 0.05$ for the PCMCI algorithm.

We compute *structural hamming distance* (SHD) and *area under the precision-recall curve* (APRC) to measure the discovered causal DAG quantitatively. SHD is a standard distance metric that compares acyclic graphs based on the counts of the edges that do not match [227,376]. It computes the difference between the two binary adjacency matrices so that missing or false edges are counted as mistakes. SHD counts two errors for a directed link with the reversed edge—for falsely directing the edge and for missing the correct edge.

$$SHD(\mathcal{G},\mathcal{H}) \leftarrow \mathcal{N}(i,j) \;\in\; V|\ \mathcal{G}(\text{edge}(i,j)) \neq \mathcal{H}(\text{edge}(i,j)) \tag{6.22}$$

where V is the set of vertices or nodes of the $\mathcal{G}$ and $\mathcal{H}$ graphs, and $\mathcal{N}$ is the number of mismatched nodes between the $\mathcal{G}$ and $\mathcal{H}$. APRC is a classification metric that evaluates predictions with a confidence score of the area under the curve of the *precision* (P) and *recall* (R) coordinates; the P and R are defined as:

$$\begin{aligned} P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN} \end{aligned} \tag{6.23}$$

where TP, FP, and FN stand for true positive, false positive, and false negative, respectively.

The sparse data handler—compressing long uniform regions of sparse binary data without anomaly status change across sensors—reduces the data from around 400K to 900 samples. This process reduces the data size by 99.76%—significantly alleviating the computational cost of the causal graph learning algorithm—PCMCI [20,236]. The compression takes roughly 8 seconds and it squeezes the uniform regions to ten samples—twice of the $\tau_{\max}$ of the CD (see Fig. 6.18c).

Fig. 6.19 illustrates the temporal GCM structure on the time-lag of $t = 0, \ldots, \tau_{\max}$—captured by the PCMCI with positive partial correlation CI test and

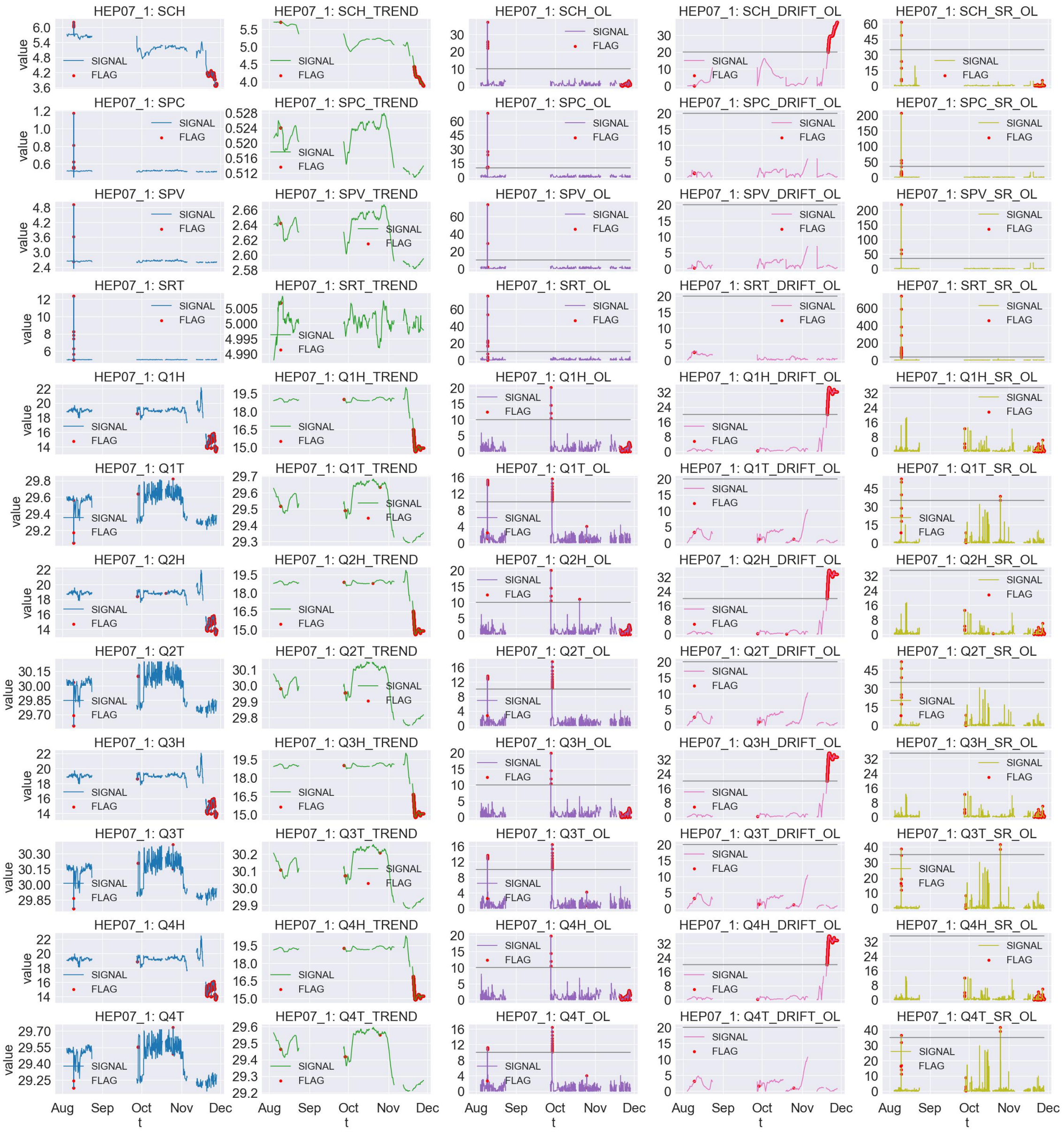


Figure 6.16: Online temporal AD on all RBX-HEP07-RM-1 sensors.

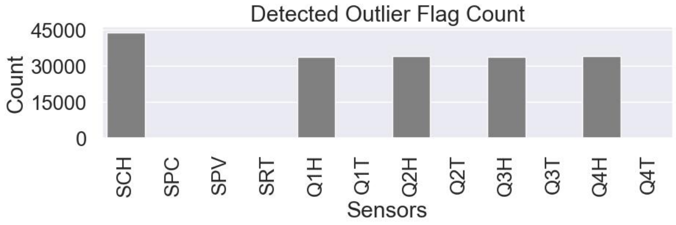
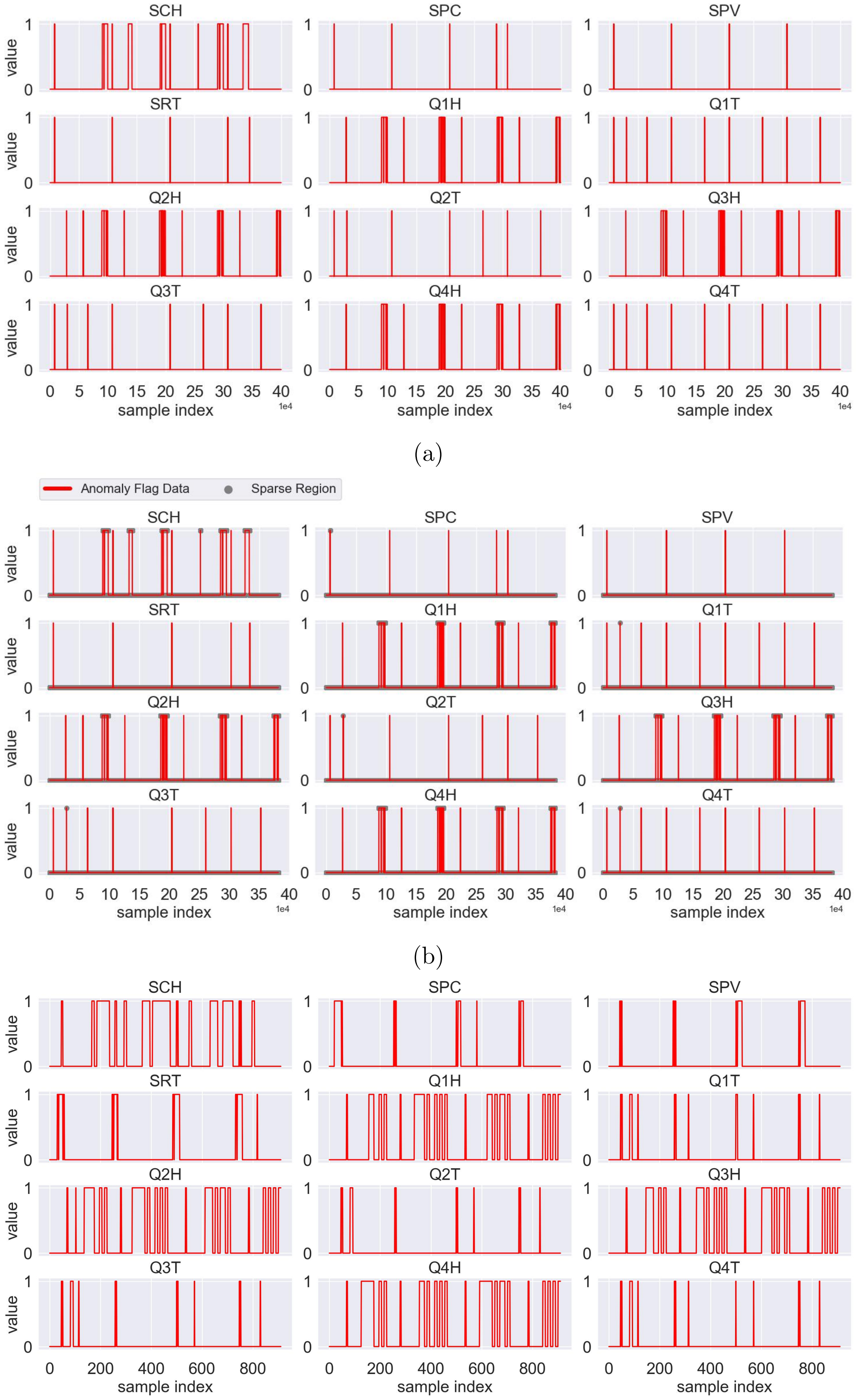


Figure 6.17: Number of detected anomaly flags from all RMs of RBX-HEP07. The humidity sensors have a higher count due to drifting trends.

imposed link assumption on time-lags. The GCM shows expected interconnection among the sensors—the clustering of environmental temperature and humidity sensors, and the link between temperature regulator Peltier voltage and current and the corresponding temperature sensors. Several bi-directed edges with multiple time-



(a)

(b)

(c)

Figure 6.18: Anomaly binary flag data from our proposed online AD approach on RBX-HEP07 sensors: a) the raw anomaly data with around 400K samples, b) the sparse regions are annotated, and c) sparse compressed data using our sparse handler algorithm—around 900 samples.

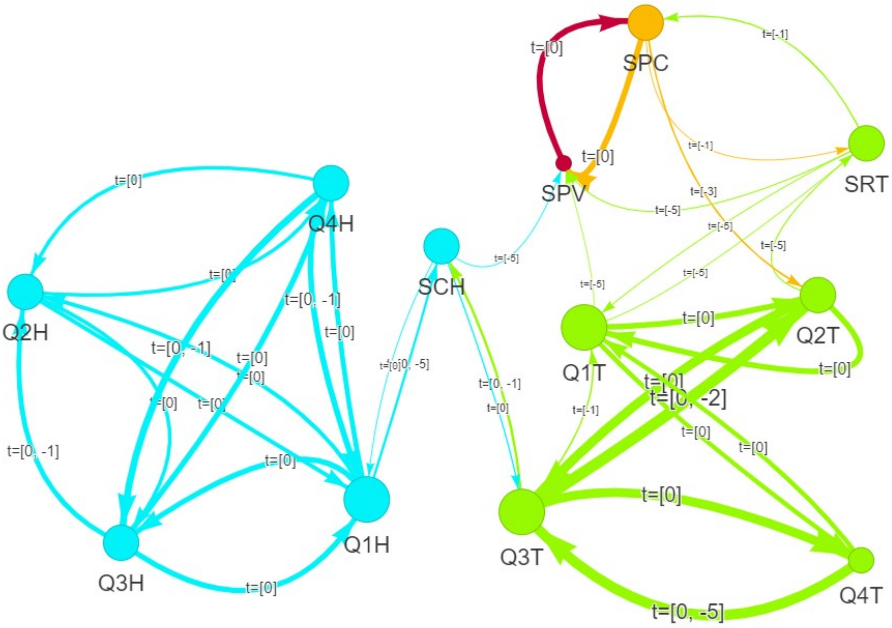


Figure 6.19: Temporal GCM of HEP07-RM using time-lag $t = 0, \ldots, 5$.

lags are also present in the network. Fig. 6.20 shows the final temporal DAG after applying Algorithm 6 the pruning. An interesting time-lagged causality link can be observed, where a temperature anomaly in SRT leads to anomalies in SPV and SPC; this is because the Peltier regulator responds—by increasing SPV and SPC—to the upsurge on the SRT.

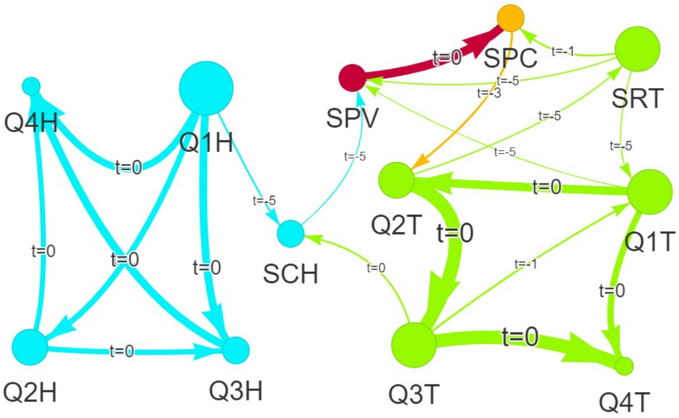


Figure 6.20: Temporal GCM-DAG network of HEP07-RM after edges pruning.

We have also evaluated the performance of causal structure learning while varying the sparse data compression length l_m. We have compared the graph accuracy and learning computational time at $l_m = 10$ with those at $l_m \in \{15, 20, 25, 30\}$ before and after pruning is applied (see Table 6.4). We conducted our experiment on a Windows 10 system with an Intel i5-8265U CPU @ 1.60 GHz (8 CPUs) and 16 GB RAM. The partial DAGs—from PCMCI with spurious links at multiple time lags—lead to higher mismatches among the PDAGs—resulting in lower APRC and higher SHD. The matching has improved when we compare the DAGs with pruned links.

The closeness of the captured graphs also demonstrates the effectiveness of causality learning on binary anomaly data using the sparse handler algorithm. We will provide further performance discussion on simulated binary data using known causal links in Section 6.2.5.

Table 6.4: Causal graph learning comparison on different sparsity length l_m. SC_t is the computation time of the PCMCI skeleton structure learning.

l_m	**Compressed Data Size**	SC_t **(sec)**↓	**Link Pruning Adjustment**	**APRC**↑	**SHD**↓
10	911	18.969	False	-	-
			True	-	-
15	1274	21.625	False	0.741	29
			True	**0.748**	**12**
20	1629	30.219	False	**0.731**	31
			True	0.723	**14**
25	1974	30.703	False	0.692	35
			True	**0.748**	**12**
30	2319	34.625	False	0.711	38
			True	**0.777**	**11**

The **black bold font** is the best score. Downarrow means lower is better, and vice versa for uparrow.

We trained our BN model—as a temporal anomaly causal inference engine shown in Fig. 6.12—using TS unrolling data and the captured anomaly causality DAG skeleton. Table 6.5 presents the probabilistic anomaly causality inference results using the trained BN. We quantify causality by calculating the anomaly conditional probability (CP) of the causes or affected sensors. The CP of anomaly occurrence for the Q1T sensor increases from 0.05 with no other evidence to above 0.90 with the evidence of a detected anomaly flag on the related sensors, Q[2-4]T, at a time-lag $t = 0$. The CP of an anomaly on the Q1H sensor increases from 0.26 to above 0.85 when there is evidence of a detected anomaly flag on the Q[2-4]H sensors at a time-lag of $t = 0$. The Q1H has a higher CP with no other evidence scenario due to trend drifts. We notice the few sample differences during drift detection across the Q[1-4]H sensors lower the causality dependency strength. The CP of SPC increases to 0.45 when SPV has an anomaly at $t = 0$. The SRT is causal to the SPC at a time-lag $t = -1$ directly and at $t = -5$ through SPV, but with uneven strength of 0.32 and 0.15, respectively. The CP rises to 0.95 on the SPC when anomalies are detected on both the SPV and SRT, indicating the CP is influenced by edge weight strength from multiple causal nodes.

The BN inference has generally produced CPs that are aligned with the link strength of the causal DAG—given in Fig. 6.20. But, caution should be taken when approaching the BN causality interpretation: 1) bidirectional edges between Q[1-4]T and Q[1-4]H indicate the presence of confounding variables—causal sufficiency assumption is not held; the temperature and humidity anomalies are externally induced (affecting the closely placed QIE cards together), and the BN inference depicts the anomaly relationship rather than the causality between the sensors, and 2) the CP—whether it is causal or influenced by observed evidence—must be explained with link edge direction; for example, the CP increase in the SPC is due to the observed anomaly on the causal SPV node, whereas the increase in the SPV is due to the observed anomaly on the influenced SPC node. Causal effect methods, such as Pearl's framework and optimal adjustment, can further be employed for causality study with intervention distribution $P(Y = 1 \mid do(X = 1))$ [377, 378].

Table 6.5: Anomaly conditional probability based on Bayesian causality inference.

Target Variable (A) at $t = 0$	**Observed Variables (B)**	$\mathcal{P}(A = 1 \mid B = 1)$
Q1T	-	0.051
	Q2T ($t = 0$)	0.904
	Q[2-4]T ($t = 0$)	0.927
Q1H	-	0.262
	Q2H ($t = 0$)	0.846
	Q[2-4]H ($t = 0$)	0.913
SPC	-	0.071
	SPV ($t = 0$)	0.456
	SRT ($t = -1$)	0.321
	SRT ($t = -5$)	0.153
	SPV ($t = 0$), SRT ($t = -1$)	0.947
SPV	-	0.057
	SPC ($t = 0$)	0.366

6.2.4 Experiment Results on Real Public Data

We present the performance of our AnomalyCD approach on a publicly available dataset[1]. The dataset is provided by EasyVista[2] and it consists of eight TS variables collected from an information technology (IT) monitoring system with a one-minute sampling rate (see Figure 6.21): 1) PMDB variable represents the extraction of some information about the messages received by the Storm ingestion system; 2) MDB refers to an activity of a process that orients messages to another process with respect to different types of messages; 3) CMB represents the activity of extraction of metrics from messages; 4) MB represents the activity of insertion of data in a database; 5) LMB reflects the updates of the last values of metrics in Cassandra; 6) RTMB represents the activity of searching to merge of data with information coming from the check message bolt; 7) GSIB represents the activity of insertion of historical status in database; 8) ESB represents the activity of writing data in Elasticsearch. EasyRCA [208] has used the dataset—index from 45683 to 50000—with anomaly regions for root cause discovery study using a prior known causal graph network of the sensors (see Fig. 6.21); each of the TS is considered anomalous with collective anomalies that have the same time of appearance and duration of 100.

We generated the binary anomaly flags dataset using our online AD approach before estimating the GCM of the EasyVista IT system. Fig. 6.22 and Fig. 6.23

[1] Available at https://github.com/ckassaad/EasyRCA

[2] https://www.easyvista.com/fr/produits/ev-observe

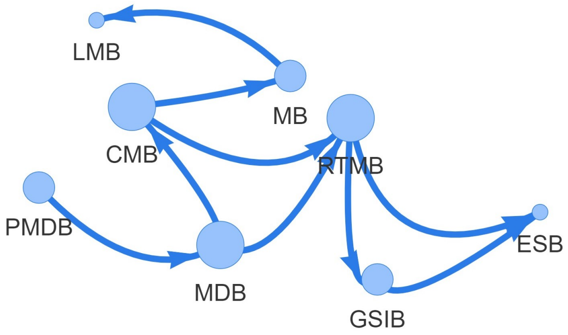


Figure 6.21: Causal graph of EasyVista's IT monitoring system during normal operation.

present the AD on TS signals and the generated binary flags, respectively. We have utilized a low threshold $\alpha_\eta = 2$ to detect more outlier noise beyond the main collective anomaly at indices 46683 to 46783; incorporating the noise outliers improves the CD since the collective anomaly affects all TS variables at the same time, and limits the causality learning from the binary data. The generated binary data contains roughly 4300 samples for each variable, and Fig. 6.24 provide the anomaly flag count; the PMDB and ESB variables generate the highest anomaly flags, and the EasyVista experts consider these two variables to be the root causes of the anomalies [208].

We utilize the causal graph of the normal operation—given in Fig. 6.21—as a reference graph to evaluate the accuracy of the estimated GCMs. We employ additional metrics for the evaluation to compare the performance of several existing CD approaches:

$$\begin{aligned}
&P = \frac{TP}{TP + FP}\\
&R = \frac{TP}{TP + FN}\\
&F_1 = \frac{2 \times P \times R}{P + R}\\
&FPR = \frac{RV + FP}{TN + FP}\\
&SHDU = UE + UM + RV
\end{aligned} \tag{6.24}$$

where the F_1 is F1-score and FPR is a false positive rate. The SHDU is undirected SHD that penalizes once—instead of twice as SHD in (6.22)—for edges with wrong directions [379]. The TP, TN, FP, and FN stand for true positive, true negative, false positive, and false negative, respectively. The RV, UE, and UM denote reversed edge, undirected extra edge, and undirected missing edge, respectively. TP is the number of edges estimated with a correct direction; TN is the number of edges that are neither in the estimated graph nor in the true graph; FP is the number of edges that are in the estimated graph but not in the true graph; FN is the number of edges that are not in the estimated graph but in the true graph; RV is the number of edges estimated with a reversed direction.

Table 6.6 presents some of the widely used CD algorithms in the literature and

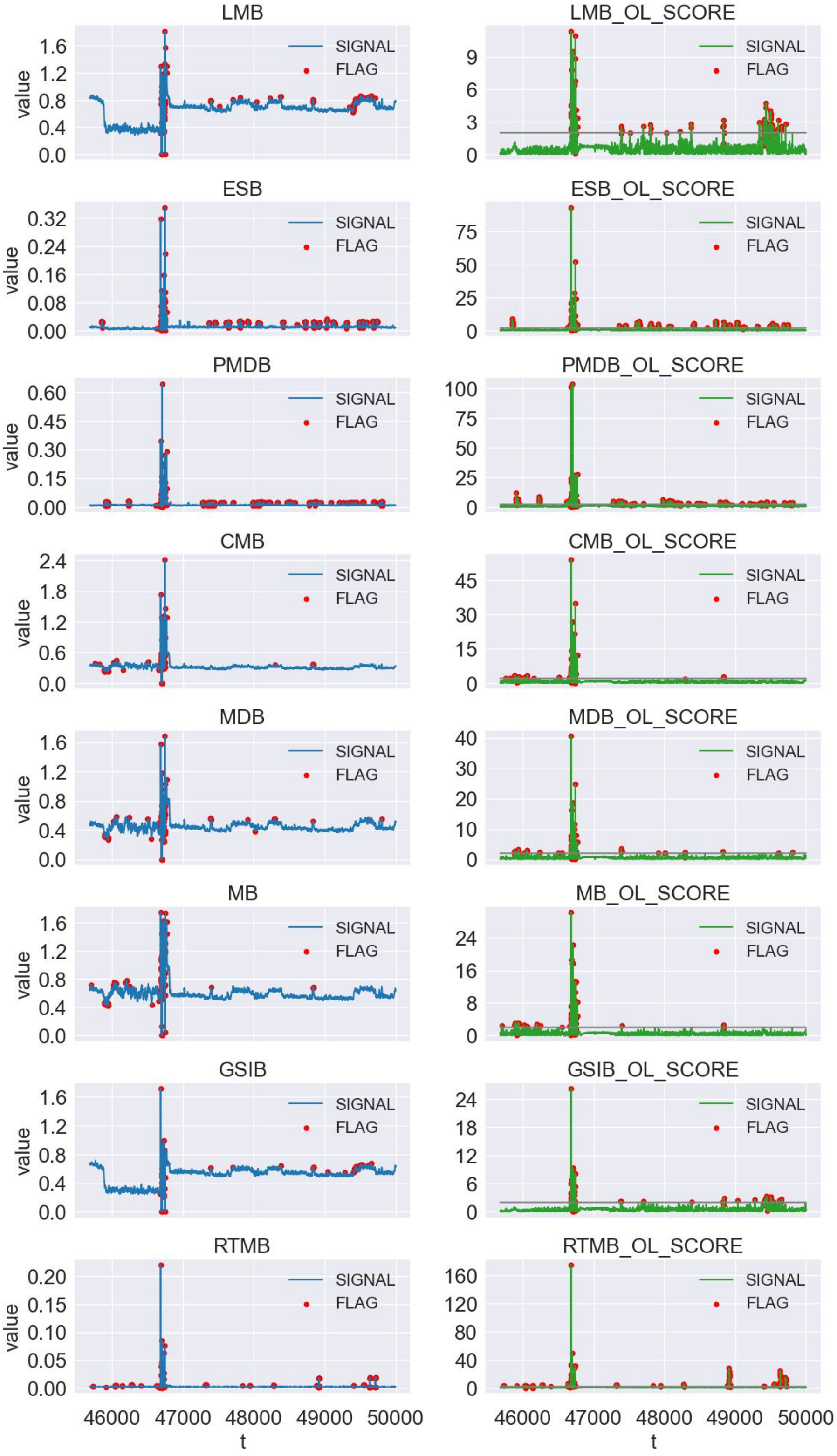


Figure 6.22: Online AD on EasyVista dataset. We employ the flags from the SR algorithm using $q_\eta = 480$ and $\alpha_\eta = 2$, as the dataset does not exhibit trend drift anomalies.

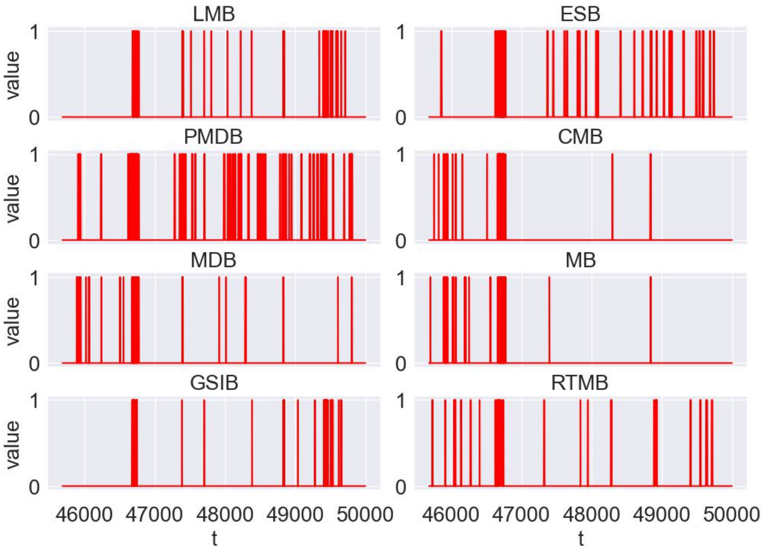


Figure 6.23: The generated binary anomaly flags of the EasyVista dataset.

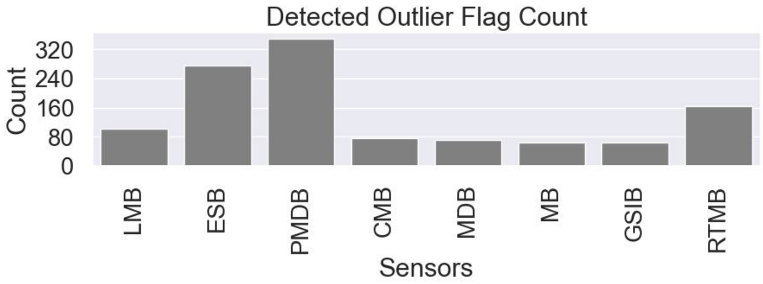


Figure 6.24: Number of detected anomalies of the EasyVista dataset.

employed in our study for comparison. We have preserved the undirected edges as bidirected for a fair comparison among the methods since some of the methods generate PDAG. We convert the temporal GCM results into summary GCM for the PCMCI-based temporal CD since the evaluation reference graph does not contain temporal information; we aggregate the time lag attributes of the edges into $t = 0$. We utilize partial correlation for the CI testing and $p_v = 0.05$ for all the constraint-based methods. We employ *Bayesian information criterion* (BiC) and *Bayesian structure scoring with Dirichlet priors* (BDeu) scores [380] for the score-based algorithms.

Table 6.6: Causal discovery methods.

Method Category	**List of Methods**
Constraint-based	PC [226], GS [223], IAMP [224], MMPC [225], PCMCI [20]
Score-based	HC [229], GES [230]
Hybrid-based	MMHC [227]
Function-based	Direct-LiNGAM [381], ICA-LiNGAM [382]
Gradient-based	GraN-DAG [383], GOLEM [384], GAE [385], RL-BIC [386], CORL [387]

Table 6.7 and Table 6.8 provide the CD performance on the raw binary anomaly

data and compressed data by sparse handling algorithm, respectively. Most methods have reached higher recalls with lower precision scores due to the bi-directed edges. Only the graph autoencoder (GAE) [385] has succeeded from the deep learning models; the RL-BIC [386] and CORL [387] employs reinforcement learning and have missed all relevant edges that require a wider search for optimal hyper-parameters to improve the performance. The sparse handler reduces the input data size by 55% using $l_m = 10$, which leads to lower computation. It also decreases the number of estimated edges, which improves the accuracy of the causal graph; it increases the F_1 score by improving the precision and reducing the false edges—decreasing the SHD and SHDU. The score-based HC [229] and its hybrid MMHC [227] algorithms have not provided accuracy leverage on the compressed binary data. Our sparse handling method achieves an average improvement of, excluding the HC and MMHC algorithms, 18%, 22%, and 15% in the F_1, FPR, and SHDU, respectively. Our AnomalyCD leads the performance in most metrics and is ranked first by the Nemenyi ranking diagram [19] over the rank scores across the seven metrics (see Fig. 6.25). The CD accuracy is not very high overall across the different approaches; the evaluation reference system graph is derived from the normal operation, and the anomaly GCM—built from the anomaly data—may behave differently.

Table 6.7: Causal graph learning on EasyVista dataset without sparse data handling.

Metric	**P↑**	**R↑**	**F_1↑**	**FPR↓**	**APRC↑**	**SHD↓**	**SHDU↓**
PC [226]	0.118	0.222	0.154	0.790	0.225	34	18
GS [223]	0.174	**0.889**	0.291	0.790	**0.539**	56	16
IAMB [224]	0.174	**0.889**	0.291	0.790	**0.539**	56	16
MMPC [225]	0.174	**0.889**	0.291	0.790	**0.539**	56	16
HC-BicScore [229]	0.167	0.222	0.191	0.526	0.249	45	13
HC-BdeuScore [229]	**0.222**	0.444	0.296	0.737	0.372	**21**	17
GES-BicScore [230]	0.160	0.444	0.235	0.895	0.341	35	18
GES-BdeuScore [230]	0.191	0.444	0.267	0.526	0.357	48	15
MMHC [227]	0.182	0.222	0.200	0.474	0.257	44	**12**
Direct-LiNGAM [381]	0.167	0.444	0.242	1.050	0.345	37	21
ICA-LiNGAM [382]	0.160	0.444	0.235	1.105	0.341	33	22
GOLEM [384]	0.167	0.222	0.191	0.526	0.249	35	16
GraN-DAG [383]	**0.222**	0.222	0.222	**0.368**	0.277	38	13
GAE [385]	0.177	0.333	0.231	0.421	0.302	47	14
PCMCI [20]	0.182	**0.889**	0.302	0.895	0.543	50	17
AnomalyCD (ours)	0.212	0.778	**0.333**	0.684	0.511	48	13

We present an ablation study in Table 6.9 for the AnomalyCD approach to demonstrate the efficacy of the additional complexity using ANAC, the sparse handling and temporal edge pruning methods leveraging the PCMCI [20]. Our sparse handling approach has enhanced the CD by 20% in the F_1 score. The P, FPR, and SHDU are substantially improved by 37%, -47%, and -41%, respectively, demonstrating improvement in the link detection accuracy; the performance decrease on the SHD by 7% relative to AnomalyCD** is due to the slight accuracy drop in

Table 6.8: Causal graph learning on EasyVista dataset with our sparse data handling method. The Δ^{+}_{avg} and Δ^{-}_{avg} are the average relative gains (compared to the performance on the raw data, given in Table 6.7) of our sparse data handling method over all the CD algorithms, including and excluding the low-performing HC and MMHC algorithms, respectively.

Metric	**P↑**	**R↑**	**F$_1$↑**	**FPR↓**	**APRC↑**	**SHD↓**	**SHDU↓**
PC [226]	**0.267**	0.444	0.333	0.474	0.395	29	12
GS [223]	0.191	0.444	0.267	0.632	0.357	47	15
IAMB [224]	0.191	0.444	0.267	0.632	0.357	47	15
MMPC [225]	0.200	**0.667**	0.308	0.474	0.457	54	12
HC-BicScore [229]	0.091	0.111	0.100	0.526	0.164	35	15
HC-BdeuScore [229]	0.111	0.222	0.148	0.842	0.221	32	19
GES-BicScore [230]	0.227	0.556	0.323	0.632	0.423	36	14
GES-BdeuScore [230]	0.191	0.444	0.267	0.526	0.357	48	15
MMHC [227]	0.100	0.111	0.105	0.474	0.168	47	14
Direct-LiNGAM [381]	0.167	0.333	0.222	0.790	0.297	38	17
ICA-LiNGAM [382]	0.235	0.444	0.308	0.684	0.379	**22**	16
GOLEM [384]	0.167	0.222	0.191	0.526	0.249	35	16
GraN-DAG [384]	0.333	0.333	0.333	**0.316**	0.380	36	12
GAE [385]	0.200	0.333	0.250	0.421	0.314	41	13
PCMCI [20]	0.207	**0.667**	0.316	0.632	0.460	52	13
AnomalyCD (ours)	0.250	**0.667**	**0.364**	0.474	**0.482**	44	**10**
Δ^{+}_{avg} (%)	**12.72**	**-10.26**	**5.80**	**17.03**	**-4.69**	**3.97**	**9.81**
Δ^{-}_{avg} (%)	**26.47**	**-1.08**	**18.31**	**22.05**	**2.64**	**7.74**	**15.45**

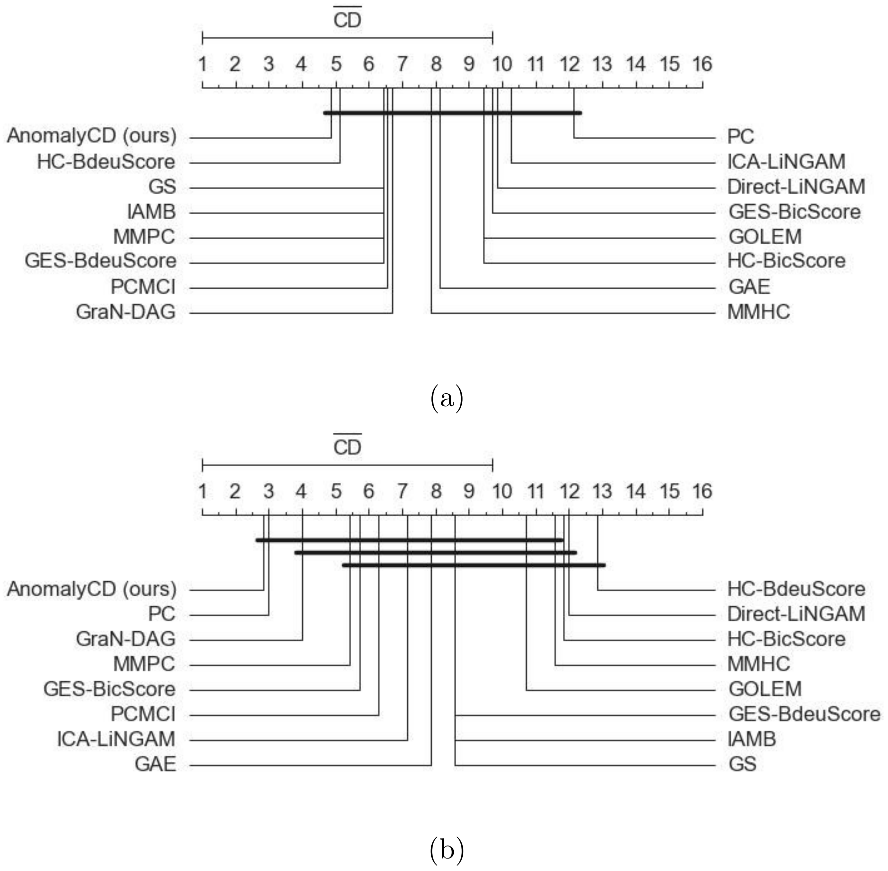
The **green** and **red bold fonts** represent an increase and decrease in performance, respectively.

the direction estimation of the bi-directed edges at $t = 0$ (see Fig. 6.26). The correlation-based CI test may remain symmetric and unable to distinguish edge direction at $t = 0$ when there is no time-lagged factor, see Eq. (6.21). The AnomalyCD-Directed refers AnomalyCD with updated edges using chi-square test for the bi-directed edges—line 17 in Algorithm 6 (fig. 6.26b). The AnomalyCD-Directed attains the best performance in most metrics, improving the CD by 83%, 26%, -53%, -36%, and -41%, in the P, F$_1$, FPR, SHD, and SHDU, respectively. The decrease in the R is because some of the bi-directed edges are removed or direction reversed by the pruning chi-square test.

6.2.5 Experiment Results on Synthetic Data

We evaluate the temporal CD of our proposed algorithms—augmenting the PCMCI algorithm—on synthetic TS anomaly data generated from known ground-truth GCM. We simulate different causal binary data for the empirical assessment with three nodes ($\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$)—representing different realistic challenges of TS binary anomaly data.

We will present the results below using latent PCMCI [228]—PCMCI with FCI



(a)

(b)

Figure 6.25: Performance ranking for pairwise comparisons using Nemenyi [19]: a) without sparse data handling, and b) with sparse data handling. The $\overline{\text{CD}}$ in the plot is the *critical difference distance*, and the horizontal bars denote mean rank differences smaller than the value of the $\overline{\text{CD}}$.

Table 6.9: Ablation study on our AnomalyCD approach using the EasyVista dataset. AnomalyCD is our temporal CD with ANAC, sparse handling, and edge pruning, AnomalyCD* is without sparse handling, AnomalyCD** is without sparse handling and edge pruning, and AnomalyCD*** is without ANAC—equivalent to the PCMCI algorithm in Ref. [20]. The AnomalyCD-Directed is AnomalyCD with directed edges.

Metric	**P↑**	**R↑**	**F_1↑**	**FPR↓**	**APRC↑**	**SHD↓**	**SHDU↓**
AnomalyCD-Directed	**0.333**	0.444	**0.381**	**0.421**	0.428	**32**	**10**
AnomalyCD	0.250	0.667	0.364	0.474	0.482	44	**10**
AnomalyCD*	0.212	0.778	0.333	0.684	0.511	48	13
AnomalyCD**	0.184	0.778	0.298	0.947	0.497	41	18
AnomalyCD*** [20]	0.182	**0.889**	0.302	0.895	**0.543**	50	17

for unseen confounding handling—due to its better accuracy for small data sets than its predecessors [20, 236]; but, it may become quite slower than the predecessors for larger data sets. We will also discuss an unrolling TS approach—inspired by the time-aware PC algorithm (TPC) [238]—to briefly demonstrate its advantages and

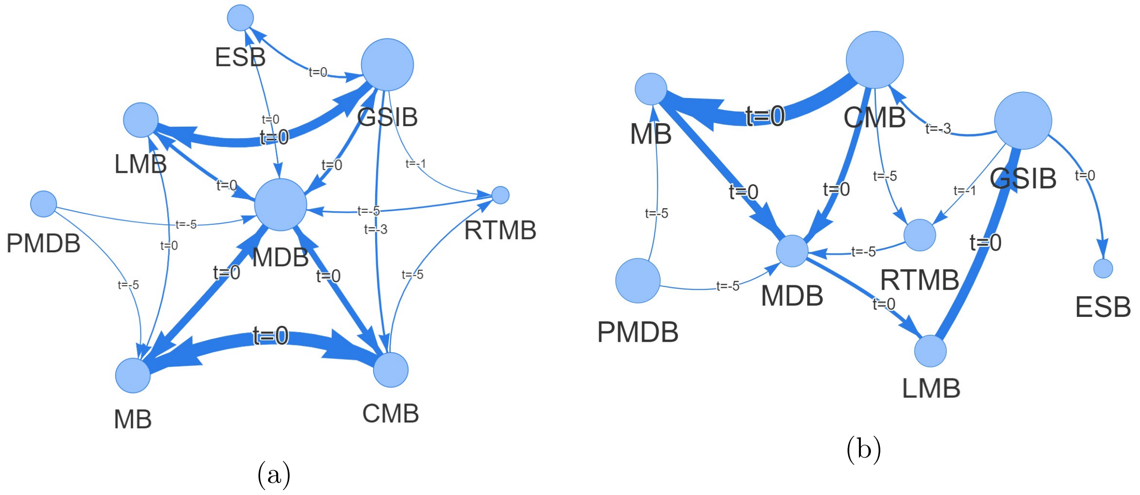


Figure 6.26: The estimated time series GCM using AnomalyCD for the EasyVista system from binary anomaly data: a) AnomalyCD, and b) AnomalyCD-Directed.

limitations compared to the PCMCI. We employ our ANAC test—using *Pearson's correlation*—for the experiments. We set the maximum causality search time-lag $\tau_{\max} = 3$ for all the experiments.

Challenge-1: Binary Anomaly Data Sparsity Compression: A binary anomaly data is synthetically generated for $\mathbf{x}_1$ with long time windows of uniform anomaly status (sparse regions), and then temporal causally dependent variables $\mathbf{x}_2$ and $\mathbf{x}_3$ are derived using $\mathbf{x}_1(t-1) \to \mathbf{x}_2(t)$ and $\mathbf{x}_2(t-3) \to \mathbf{x}_3(t)$ (see Fig. 6.27). We set the maximum time-lag $\tau_{\max} = 3$ for causality search that excludes the deduced edge $\mathbf{x}_1(t-4) \to \mathbf{x}_3(t)$.

Our proposed sparsity data handler approach enables capturing causality information from the anomaly flag transition edges by avoiding extended uniform regions (see algorithm 4). The algorithm has compressed the expended regions to substantially lessen the computation cost and improve the accuracy of the GCM discovery (see Fig. 6.28); it reduces the Type-I—false positive edges—errors (see Fig. 6.30). It decreases the data samples and the average PCMCI computation time by around 60% and 42%, respectively (see Fig. 6.28 and 6.29b). The compressed data results in correct causal graph structure APRC = 1.00 and SHD = 0) at different CI threshold p_v (see Fig. 6.29), whereas the sparse data causes spurious links $\mathbf{x}_1(t-1) \to \mathbf{x}_1(t)$ and $\mathbf{x}_2(t-1) \to \mathbf{x}_2(t)$ (see Fig. 6.30) that lower APRC and increase SHD scores to 0.83 and 1, respectively.

Challenge-2: Incomplete Stationary Binary Data: System interruption often follows after anomaly alerts for maintenance or prevention of further damage. We generated temporally incomplete anomaly data when the anomaly signals from all the variables reset to zero—simulating system interruption occurrence in the data; we utilized the binary data from the previous section—$\mathbf{x}_1(t-1) \to \mathbf{x}_2(t)$ and $\mathbf{x}_2(t-3) \to \mathbf{x}_3(t)$—and applied an interruption condition $(\mathbf{x}_1 = 0) \cap (\mathbf{x}_2 = 0) \cap (\mathbf{x}_3 =$

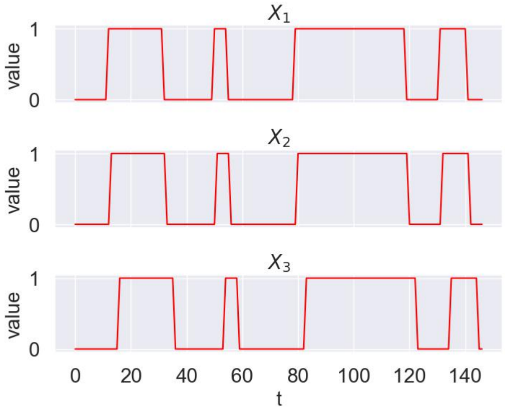
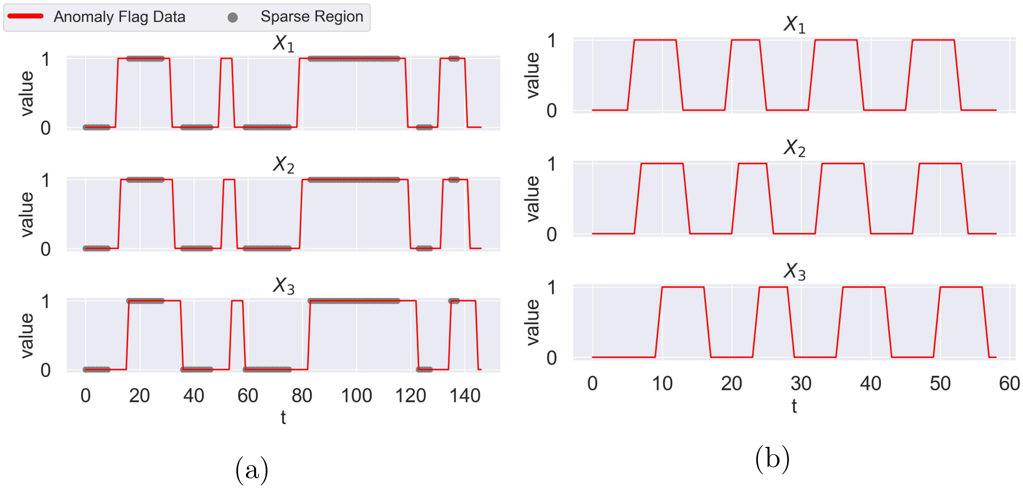


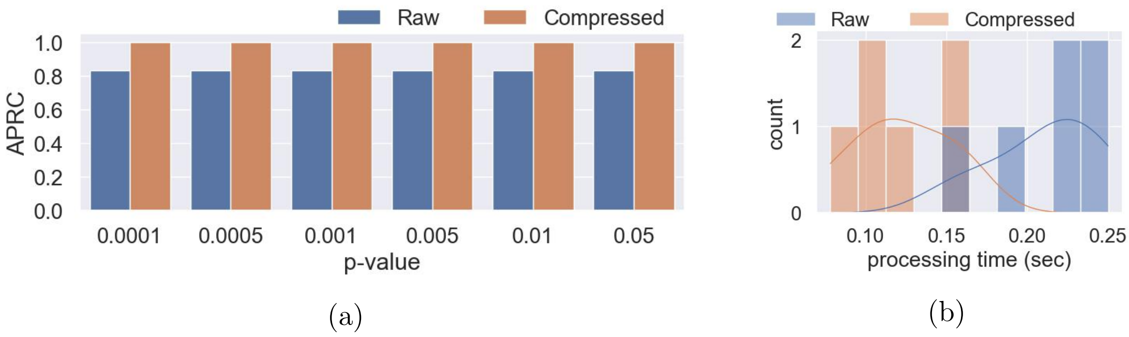
Figure 6.27: Raw binary anomaly-flag data with sparse uniform state regions. The data is generated using $\mathbf{x}_1(t-1) \rightarrow \mathbf{x}_2(t)$ and $\mathbf{x}_2(t-3) \rightarrow \mathbf{x}_3(t)$.



(a)

(b)

Figure 6.28: Data compression using our sparse data handler algorithm: a) raw TS binary anomaly flag with marked sparse regions, and b) compressed TS anomaly data after sparsity handling.



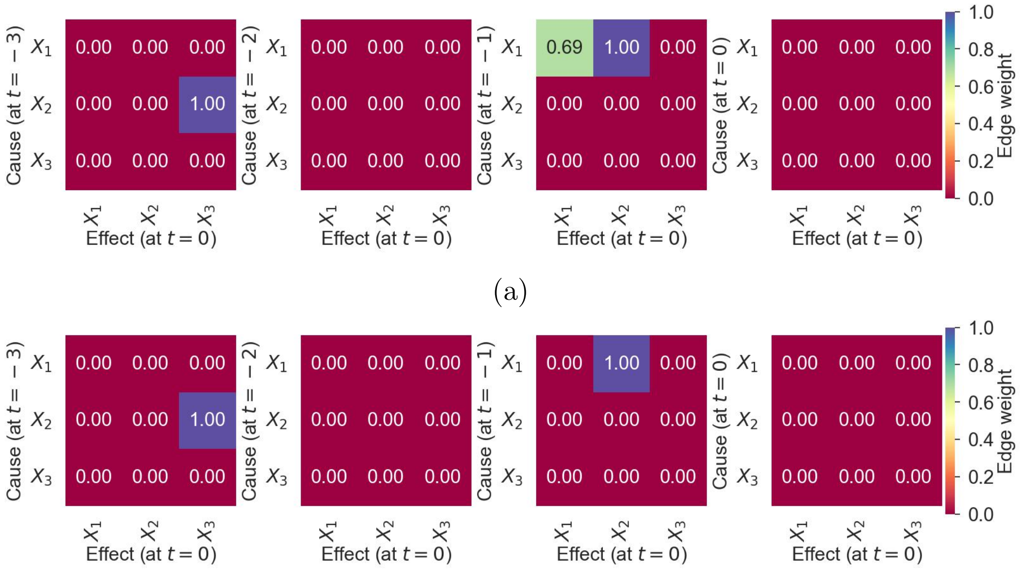
(a)

(b)

Figure 6.29: Graph discovery performance score on raw and compressed binary anomaly data. APRC = 0.83 and SHD = 1, and APRC = 1.00 and SHD = 0 for all the p_v values for the raw and compressed data, respectively.

1).

The data incompleteness greatly diminishes the edge weight between $\mathbf{x}_2(t-1) \rightarrow$

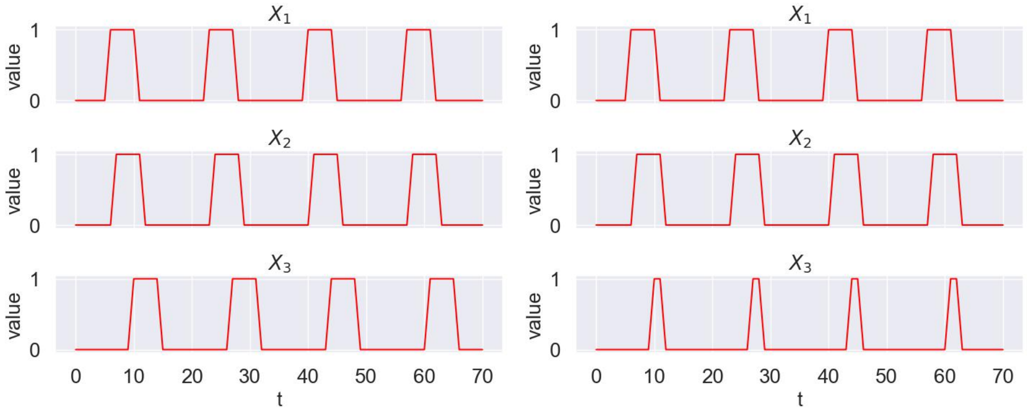


(a)

(b)

Figure 6.30: Heatmaps of the estimated temporal graph edge weights at $p_v = 0.05$ using a) raw and b) compressed binary anomaly data.

$\mathbf{x}_3(t)$ from 1.00 to 0.28, introduces spurious links at higher p_v, and removes proper edges at lower p_v (see Fig. 6.32 and Fig. 6.33).



(a)

(b)

Figure 6.31: Generated binary anomaly-flag data using $\mathbf{x}_1(t-1) \rightarrow \mathbf{x}_2(t)$ and $\mathbf{x}_2(t-3) \rightarrow \mathbf{x}_3$ and an interruption condition $(\mathbf{x}_1 = 0) \cap (\mathbf{x}_2 = 0) \cap (\mathbf{x}_3 = 1)$: a) before, and b) after the interruption. The interruption makes the temporal data of $\mathbf{x}_3$ incomplete.

Challenge-3: Incomplete Non-stationary Binary Data: Most temporal CD studies assume stationary causality. But, one of the most challenging tasks in dealing with real-world TS binary anomaly-flag data is non-stationary causality—the presence of different interactions at different times. A few studies propose mitigation strategies for non-stationarity—e.g., partitioning the data into disjoint time

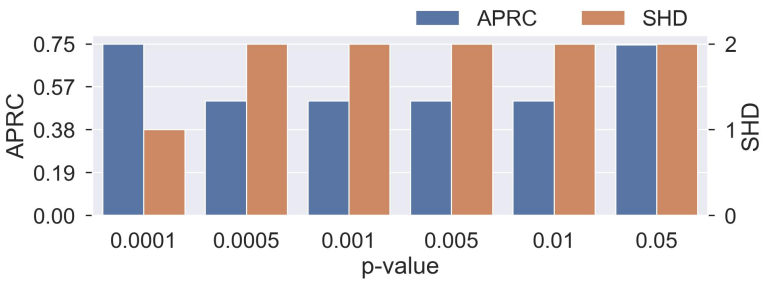
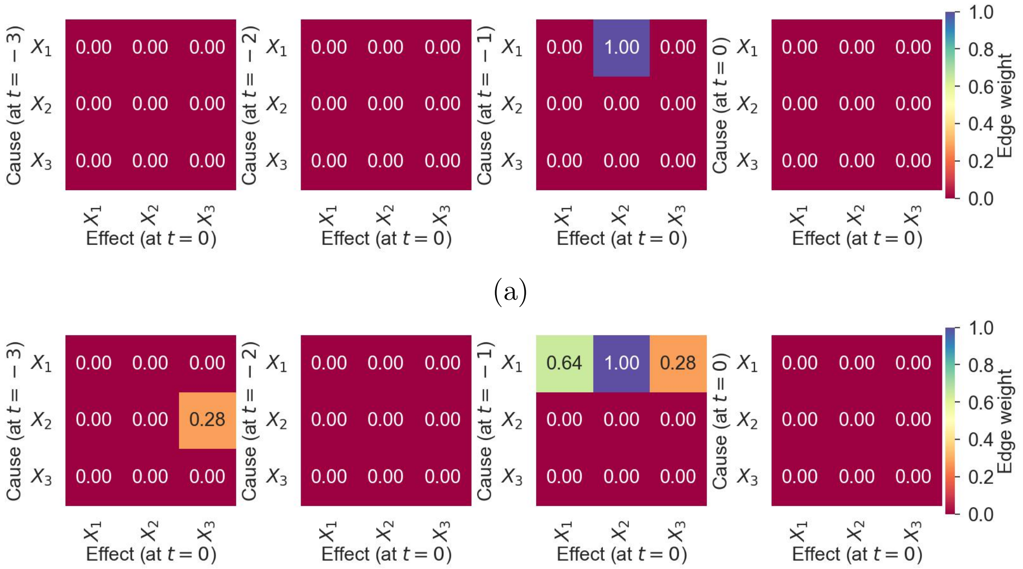


Figure 6.32: Graph discovery performance score on incomplete binary anomaly data.



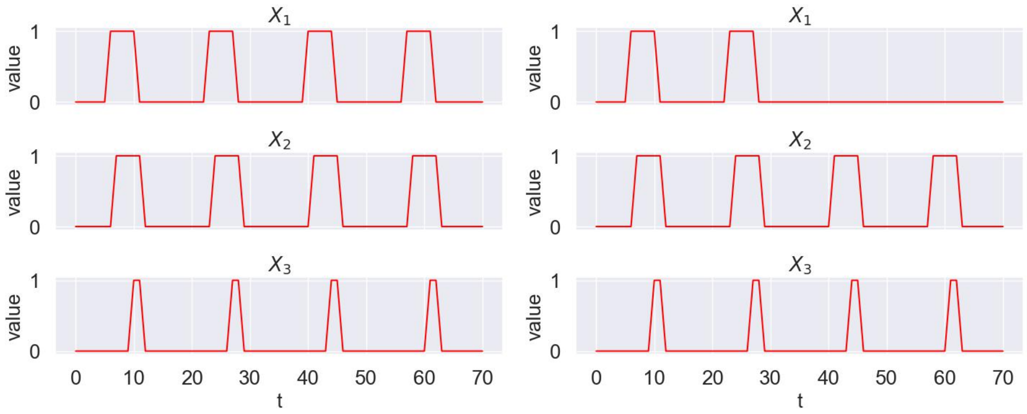
(a)

(b)

Figure 6.33: The estimated temporal GCM heatmaps on incomplete binary anomaly data: a) $p_v = 0.0001$ (APRC $= 0.77$ and SHD $= 1$), and b) $p_v = 0.05$ (APRC $= 0.75$ and SHD $= 2$).

windows and expecting each with its own causal network [237]; the time windowing omits relations across windows, and results may vary with the choice of window size [209]. We assess the impact of having non-stationary causality in anomaly data and drive guiding insights that might be relevant for tuning and building anomaly causality graphs.

We inject non-stationarity at $t > 30$ by setting $\mathbf{x}_1(t > 30) = 0$ that removing the edge $\mathbf{x}_1(t-1) \to \mathbf{x}_2(t)$ on the incomplete anomaly data—generated in the previous section (see Fig. 6.34). The discovered graphs have spurious self-time-lagged edges (APRC $= 0.75$ and SHD $= 2$ for all p_v), and the edge weights are reduced below 1.00 for the correct links (see Fig. 6.36). We leverage the PCMIC algorithm with a prior anomaly causality link assumption that excludes causality from self-time-lag edges to improve its capability: The prior link assumption has enhanced the PCMCI in detecting the causal graph accurately—except at high $p_v = 0.05$—by mitigating the false edges and increasing weights of the true edges incomplete and non-stationary binary anomaly data (see Fig. 6.37 and Fig. 6.38).



(a)

(b)

Figure 6.34: Generated temporal binary anomaly-flag data using $\mathbf{x}_1(t-1) \rightarrow \mathbf{x}_2(t)$ and $\mathbf{x}_2(t-3) \rightarrow \mathbf{x}_3(t)$ with an interruption condition $(\mathbf{x}_1 = 0) \cap (\mathbf{x}_2 = 0) \cap (\mathbf{x}_3 = 1)$ and non-stationarity injected at $t > 30$ on $\mathbf{x}_1$: a) anomaly data with the interruption condition and b) after the non-stationarity is applied on $\mathbf{x}_1$.

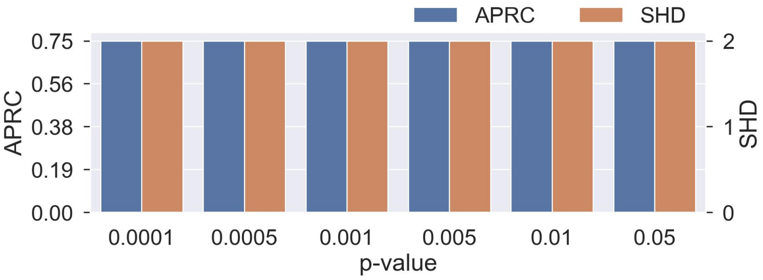
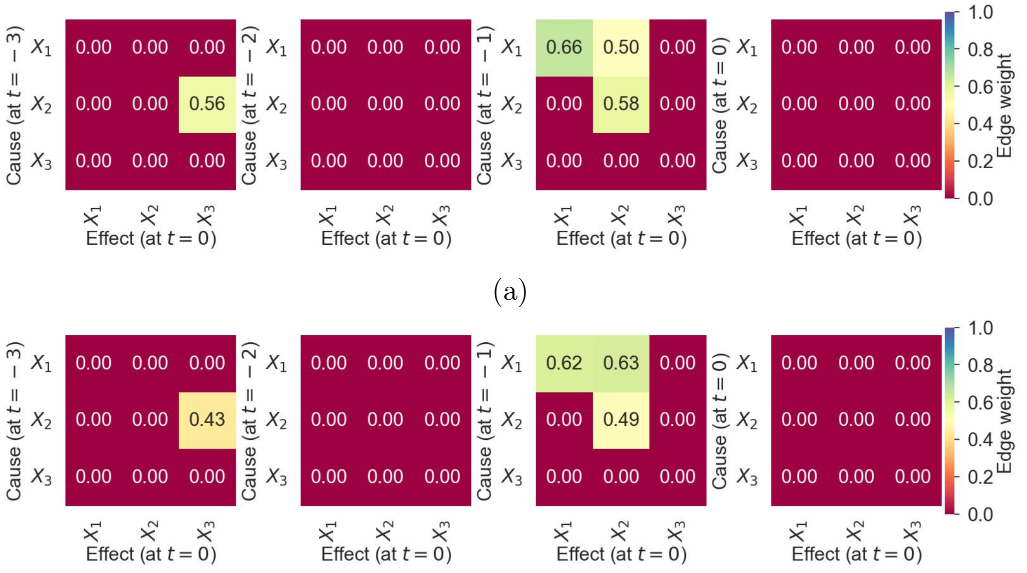


Figure 6.35: Graph discovery performance score on incomplete and non-stationary binary anomaly data.

Addressing Challenge-3 through Time Series Unrolling Mechanism: [238] proposes a promising TPC that employs unrolling the TS data—by adding new nodes with time delay tags—to generate DAG using the PC algorithm. The authors apply a set of conditions, such as avoiding backward causality in time and weight thresholding to direct edges and pruning the DAG, respectively, when rolling the DAG. We evaluate the unrolling approach with PCMCI [228] to validate its advantages on the temporal CD of binary anomaly data. We employ PCMCI instead of PC because of better accuracy with MCI; the PCMCI also enables us to compare with previous results consistently. The unrolling approach suffers from false contemporaneous and self-lagged links (see Fig. 6.40 and 6.41). The performance substantially improves with a prior anomaly link assumption; it still does not work best for the extreme p_v (see Fig. 6.42 and Fig. Fig. 6.43) We have found two major potential disadvantages of the unrolling approach for GCM learning: 1) the computation increases with the variable and time lag dimensions—limited impact on relieving computational overhead, and 2) it needs more effort in tuning the reasonable p_v than using the PCMCI directly.



(a)

(b)

Figure 6.36: The estimated temporal GCM heatmaps on incomplete and non-stationary binary anomaly data: a) $p_v = 0.0001$ (APRC $= 0.75$ and SHD=2), and b) $p_v = 0.05$ (APRC $= 0.75$ and SHD=2).

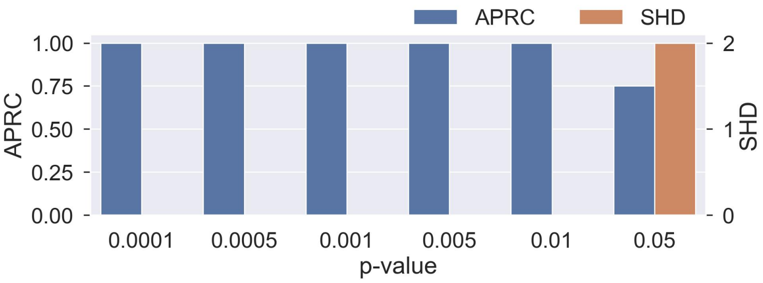
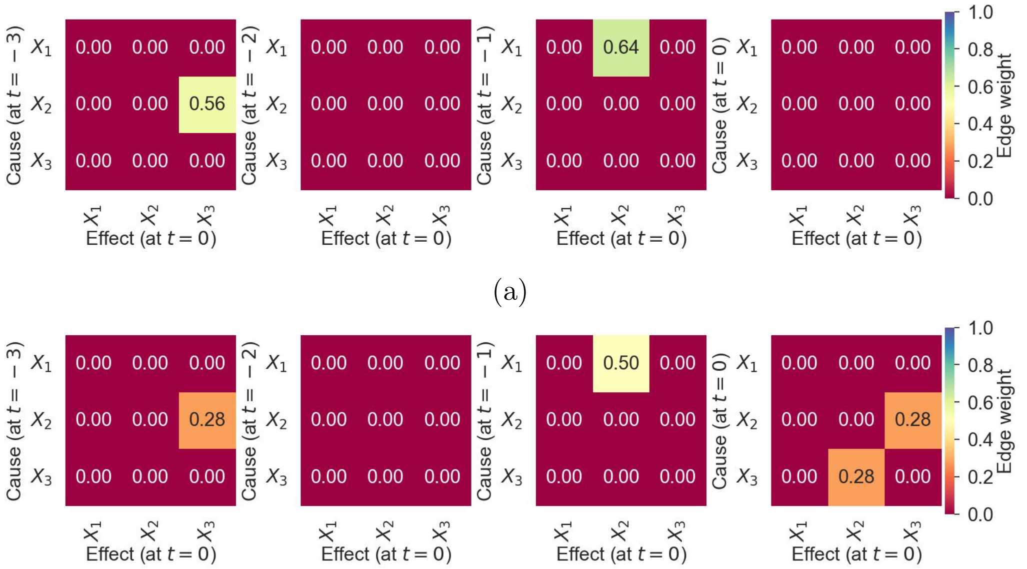


Figure 6.37: Graph discovery performance score using prior link assumption on incomplete and non-stationary binary anomaly data.

6.2.6 Summary

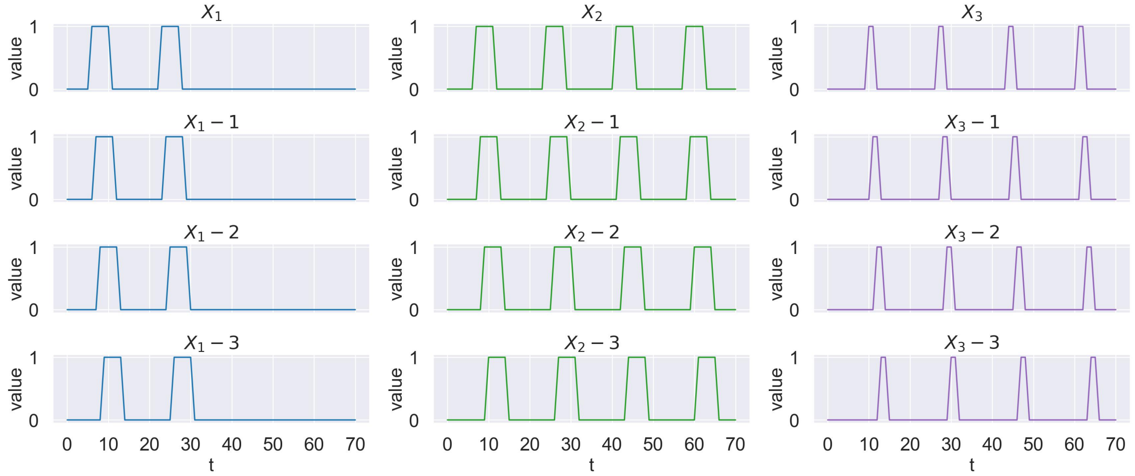
We have introduced an unsupervised framework for causal discovery on binary anomaly data. Our framework includes several algorithms and approaches to address challenges related to inferring causality in sparse anomaly binary data. The approaches have established promising accuracy in unsupervised online anomaly detection and significantly reduced the computational overhead of graph CD on binary data. Our methods achieve fast and accurate CD by employing sparse binary data compression, prior time-lag link assumptions, and edge running and adjustment. The experiment results of the Hadron Calorimeter of the CMS experiment, public IT system monitoring, and simulated causal data sets have demonstrated the efficacy of the proposed approaches. Our approaches will help facilitate diagnostic tasks across diverse systems of the CMS HCAL, where flexible methods are needed for multi-level complex system configurations and limited annotated data. We rec-



(a)

(b)

Figure 6.38: Discovered temporal GCM heatmaps using prior link assumption on incomplete and non-stationary binary anomaly data: a) $p_v = 0.0001$ (APRC $= 1.00$ and SHD $= 0$), and b) $p_v = 0.05$ (APRC $= 0.75$ and SHD $= 2$).



(a)

Figure 6.39: Unrolled binary anomaly flag time series data up to maximum time-lag $\tau = 3$. The temporal binary anomaly-flag data using $\mathbf{x}_1(t-1) \rightarrow \mathbf{x}_2$ and $\mathbf{x}_2(t-3) \rightarrow \mathbf{x}_3(t)$ with an interruption condition $(\mathbf{x}_1 = 0) \cap (\mathbf{x}_2 = 0) \cap (\mathbf{x}_3 = 1)$ and non-stationarity injected at $t > 30$ on $\mathbf{x}_1$.

ommend further breakdown of the discovered causal networks for the root-cause identification of anomalies using existing methods, such as CausalRCA and MicroCause. The target anomaly time regime must be defined before the causal graph learning since different anomalies may have different root causes.

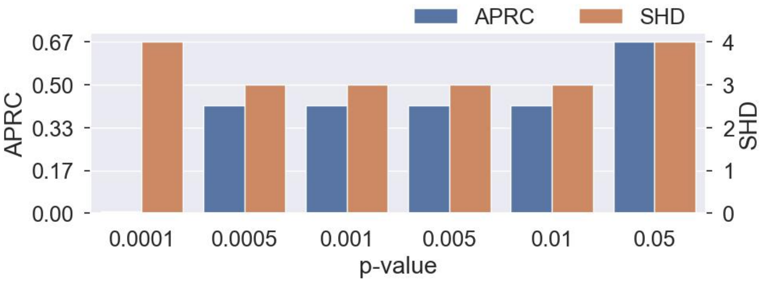


Figure 6.40: Graph discovery performance score on unrolled incomplete and non-stationary binary anomaly data.

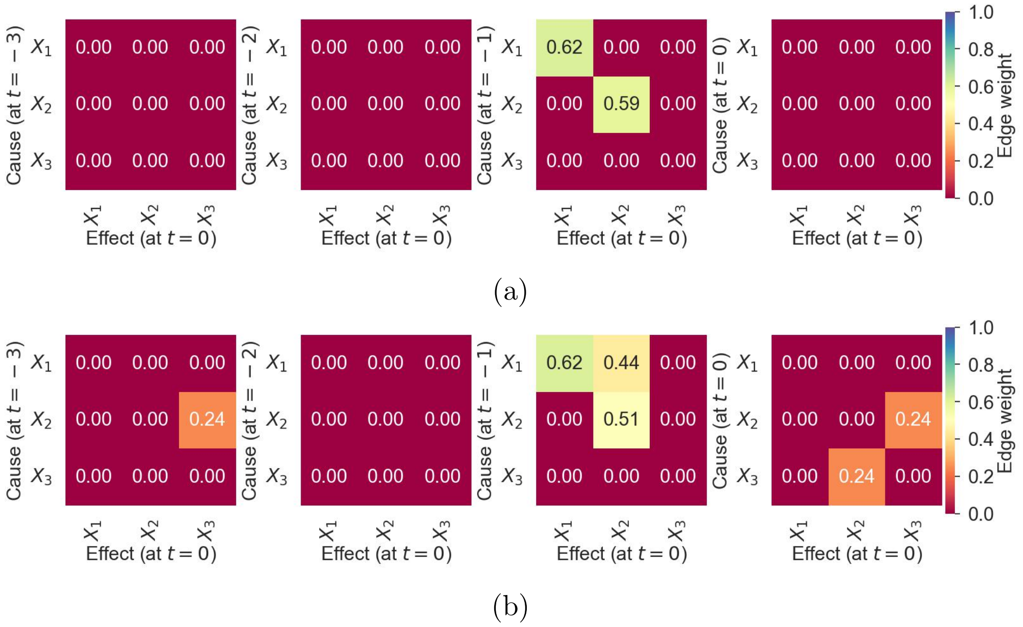


Figure 6.41: The estimated temporal GCM heatmaps on unrolled incomplete and non-stationary binary anomaly data: a) $p_v = 0.0001$ (APRC $= 0$ and SHD $= 4$), and b) $p_v = 0.05$ (APRC $= 0.67$ and SHD $= 4$).

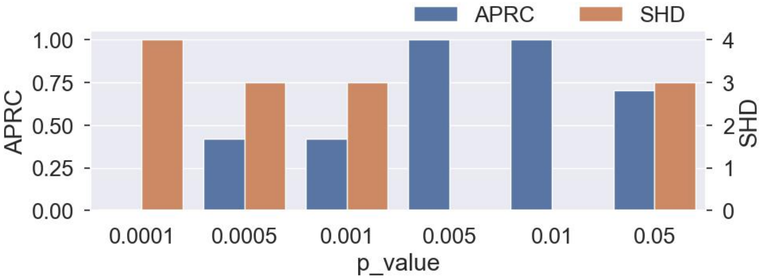
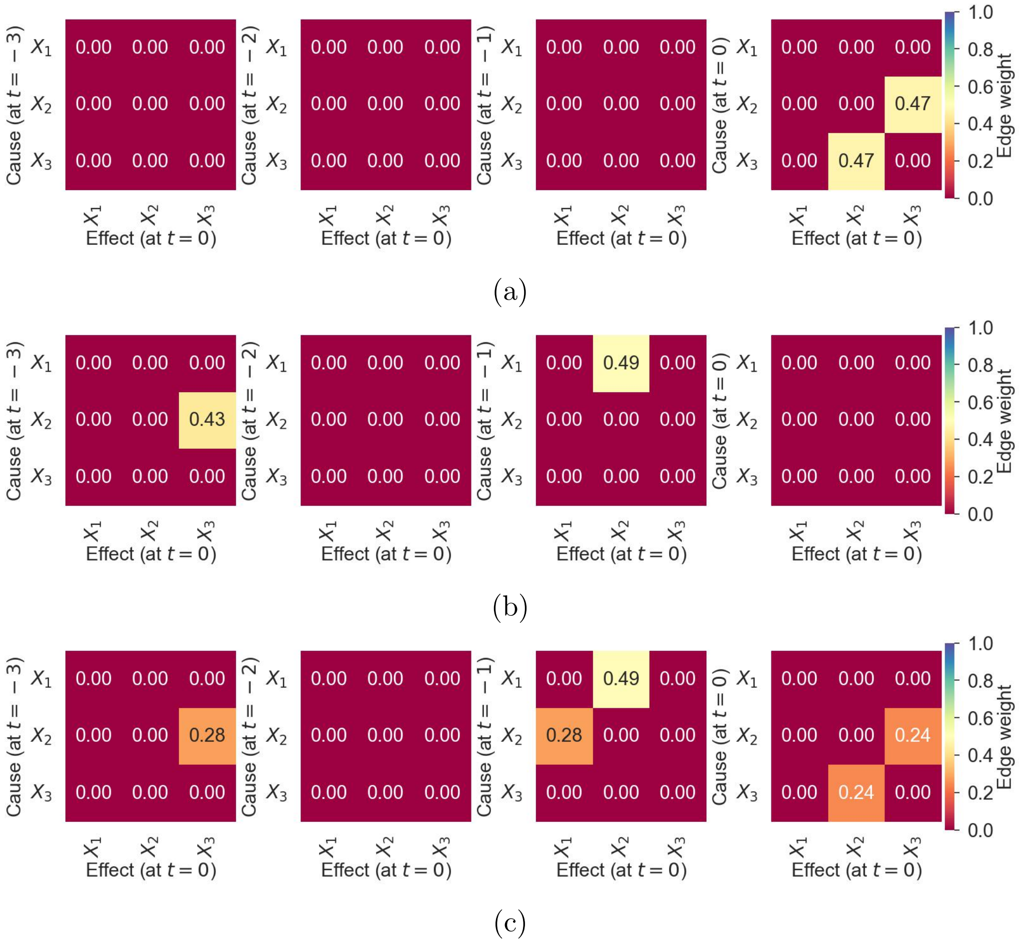


Figure 6.42: Graph discovery performance score using prior link assumption on unrolled incomplete and non-stationary binary anomaly data.



(a)

(b)

(c)

Figure 6.43: The estimated temporal GCM heatmaps using prior temporal link assumption on unrolled incomplete and non-stationary binary anomaly data: a) $p_v = 0.0001$ (APRC = 0 and SHD = 4), b) $p_v = 0.01$ (APRC = 1.00 and SHD = 0), and c) $p_v = 0.05$ (APRC = 0.70 and SHD = 3).

Chapter 7

System Integration and Deployment

This chapter will discuss the production integration of our machine learning models—discussed from Chapter 4 to Chapter 6—in the monitoring systems of CMS.

Our tools have been deployed in CMS—actively assist the HCAL operation and DQM experts in detecting and diagnosing system faults across thousands of sensors. We developed a DESMOD dashboard[1] to provide interactive access to the developed AD models and analysis tools on the data retrieved from the MonDB databases of the 904—operation investigation laboratory at CMS—and the P5—the CMS detector at the LHC (see Fig. 7.1). The DQM AD models are integrated[2] in the CMSSW production engine[3] for real-time monitoring of the data acquisition channels of HE and HB subdetectors. The online models are deployed in C++ environments for faster inference and compiled using open neural network exchange (ONNX)[4]. We provide access to the deployed tools through our DESMOD dashboard for offline fault detection and diagnostics. The offline diagnostics of the DQM AD models utilize PyROOT—a Python extension module for interacting with DQM ROOT format data [388].

7.1 Sensor Fault Detection and Diagnostics Tools

The diagnostic sensor monitoring page depicted in Fig. 7.2 incorporates several tools for offline sensor data analysis, such as the AD and AP—discussed in Section 4.1 and 5.1—and anomaly causality discovery tools—discussed in Section 6 (see from Fig. 7.6 to Fig. 7.9). It also integrates sensor data preprocessing, multi-level variable filtering, data cleaning and normalization, and interactive signal visualization and clustering tools (see from Fig. 7.3 to Fig. 7.5).

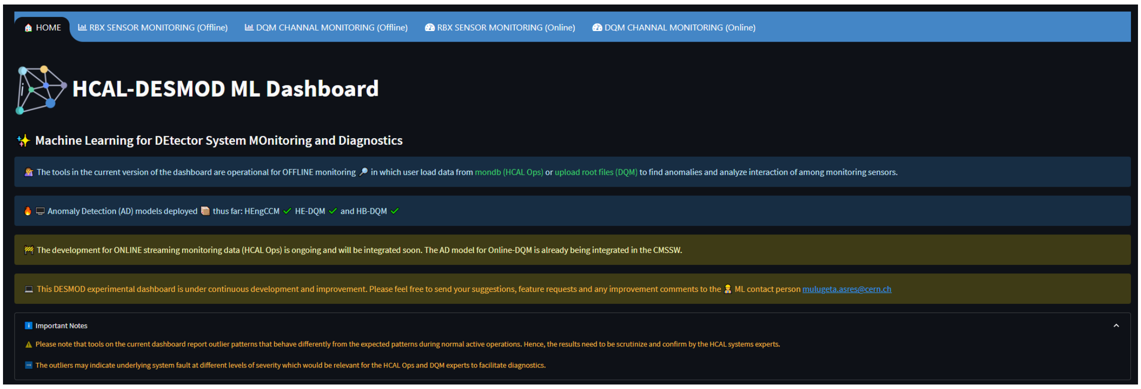
Fig. 7.3 shows the multivariate time series data preprocessing panel that includes data selection and manipulation tools, multi-system and sensor variable selection, and signal visualization. We provide temporal system operation variation detection through sensor data clustering in Fig. 7.4. Fig. 7.5 illustrates system deviation

[1] Our DESMOD dashboard portal at https://cmshcalweb01.cern.ch/desmod

[2] ML4DQM pull request to the CMSSW at https://github.com/cms-sw/cmssw/pull/42212

[3] The CMSSW core production at https://github.com/cms-sw/cmssw

[4] ONNX at https://onnx.ai


HOME
RBX SENSOR MONITORING (Offline)
DQM CHANNAL MONITORING (Offline)
RBX SENSOR MONITORING (Online)
DQM CHANNAL MONITORING (Online)

HCAL-DESMOD ML Dashboard

✨ Machine Learning for DEtector System MOnitoring and Diagnostics

The tools in the current version of the dashboard are operational for OFFLINE monitoring 🔎 in which user load data from mondb (HCAL Ops) or upload root files (DQM) to find anomalies and analyze interaction of among monitoring sensors.

🔥 💻 Anomaly Detection (AD) models deployed 📦 thus far: HEngCCM ✓ HE-DQM ✓ and HB-DQM ✓

🚧 The development for ONLINE streaming monitoring data (HCAL Ops) is ongoing and will be integrated soon. The AD model for Online-DQM is already being integrated in the CMSSW.

💻 This DESMOD experimental dashboard is under continuous development and improvement. Please feel free to send your suggestions, feature requests and any improvement comments to the ML contact person mulugeta.asres@cern.ch

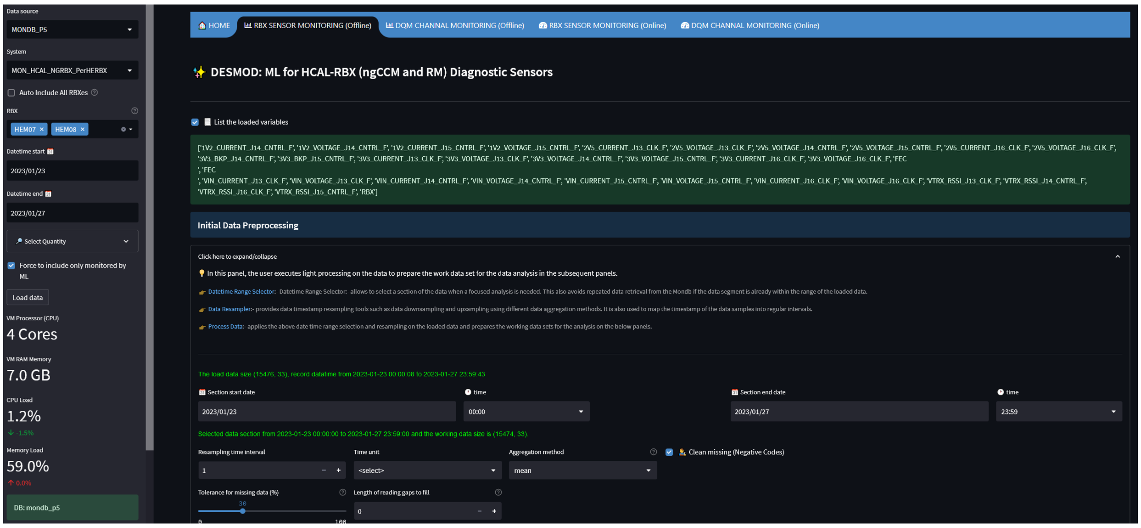
Important Notes

⚠ Please note that tools on the current dashboard report outlier patterns that behave differently from the expected patterns during normal active operations. Hence, the results need to be scrutinize and confirm by the HCAL systems experts.

The outliers may indicate underlying system fault at different levels of severity which would be relevant for the HCAL Ops and DQM experts to facilitate diagnostics.

Figure 7.1: The DESMOD dashboard—locally deployed at CMS to provide fault detection and diagnostics on the multivariate sensor reading and high-dimensional spatial data quality monitoring histograms.

clustering on multi-RBX systems using dimensional reduction embedding on multivariate sensors. Fig. 7.6 provides online time series AD using an ensemble of outlier detection algorithms without prior model training requirements to detect typical univariate TS anomalies (see Section 6.2.2). Fig. 7.7 depicts AD results from our CGVAE deep learning model for the ngCCM of the HE calorimeter—summary AD report per system and sensor (see Section 4.1) [41]. Fig. 7.8 shows a trend drifting anomaly on the RSSI sensor—one of the challenging anomalies to capture. The dashboard also provides divergence detection and causality discovery tools—discussed in Chapter 6—for the multi-system and multivariate RBX systems (see Fig. 7.9).


HOME
RBX SENSOR MONITORING (Offline)
DQM CHANNAL MONITORING (Offline)
RBX SENSOR MONITORING (Online)
DQM CHANNAL MONITORING (Online)
Select Quantity
Load data

Data source: MONDB_P5

System
MON_HCAL_NGRBX_PerHERBX

☐ Auto Include All RBXes

RBX: HEM07 × HEM08 ×

Datetime start: 2023/01/23

Datetime end: 2023/01/27

☑ Force to include only monitored by ML

VM Processor (CPU)
4 Cores

VM RAM Memory
7.0 GB

CPU Load
1.2%
↓ -1.5%

Memory Load
59.0%
↑ 0.0%

DB: mondb_p5

✨ DESMOD: ML for HCAL-RBX (ngCCM and RM) Diagnostic Sensors

☑ List the loaded variables

```
['1V2_CURRENT_J14_CNTRL_F', '1V2_VOLTAGE_J14_CNTRL_F', '1V2_CURRENT_J15_CNTRL_F', '1V2_VOLTAGE_J15_CNTRL_F', '2V5_CURRENT_J13_CLK_F', '2V5_VOLTAGE_J13_CLK_F', '2V5_VOLTAGE_J14_CNTRL_F', '2V5_VOLTAGE_J15_CNTRL_F', '2V5_CURRENT_J16_CLK_F', '2V5_VOLTAGE_J16_CLK_F',
'3V3_BKP_J14_CNTRL_F', '3V3_BKP_J15_CNTRL_F', '3V3_CURRENT_J13_CLK_F', '3V3_VOLTAGE_J13_CLK_F', '3V3_VOLTAGE_J14_CNTRL_F', '3V3_VOLTAGE_J15_CNTRL_F', '3V3_CURRENT_J16_CLK_F', '3V3_VOLTAGE_J16_CLK_F', 'FEC
', 'FEC
', 'VIN_CURRENT_J13_CLK_F', 'VIN_VOLTAGE_J13_CLK_F', 'VIN_CURRENT_J14_CNTRL_F', 'VIN_VOLTAGE_J14_CNTRL_F', 'VIN_CURRENT_J15_CNTRL_F', 'VIN_VOLTAGE_J15_CNTRL_F', 'VIN_CURRENT_J16_CLK_F', 'VIN_VOLTAGE_J16_CLK_F', 'VTRX_RSSI_J13_CLK_F', 'VTRX_RSSI_J14_CNTRL_F',
'VTRX_RSSI_J16_CLK_F', 'VTRX_RSSI_J15_CNTRL_F', 'RBX']
```

Initial Data Preprocessing

Click here to expand/collapse

💡 In this panel, the user executes light processing on the data to prepare the work data set for the data analysis in the subsequent panels.

👉 Datetime Range Selector:- Datetime Range Selector:- allows to select a section of the data when a focused analysis is needed. This also avoids repeated data retrieval from the Mondb if the data segment is already within the range of the loaded data.

👉 Data Resampler:- provides data timestamp resampling tools such as data downsampling and upsampling using different data aggregation methods. It is also used to map the timestamp of the data samples into regular intervals.

👉 Process Data:- applies the above date time range selection and resampling on the loaded data and prepares the working data sets for the analysis on the below panels.

The load data size (15476, 33), record datatime from 2023-01-23 00:00:08 to 2023-01-27 23:59:43

Section start date: 2023/01/23 time: 00:00

Section end date: 2023/01/27 time: 23:59

Selected data section from 2023-01-23 00:00:00 to 2023-01-27 23:59:00 and the working data size is (15474, 33).

Resampling time interval: 1

Time unit
<select>

Aggregation method: mean

☑ Clean missing (Negative Codes)

Tolerance for missing data (%): 30 (0–100)

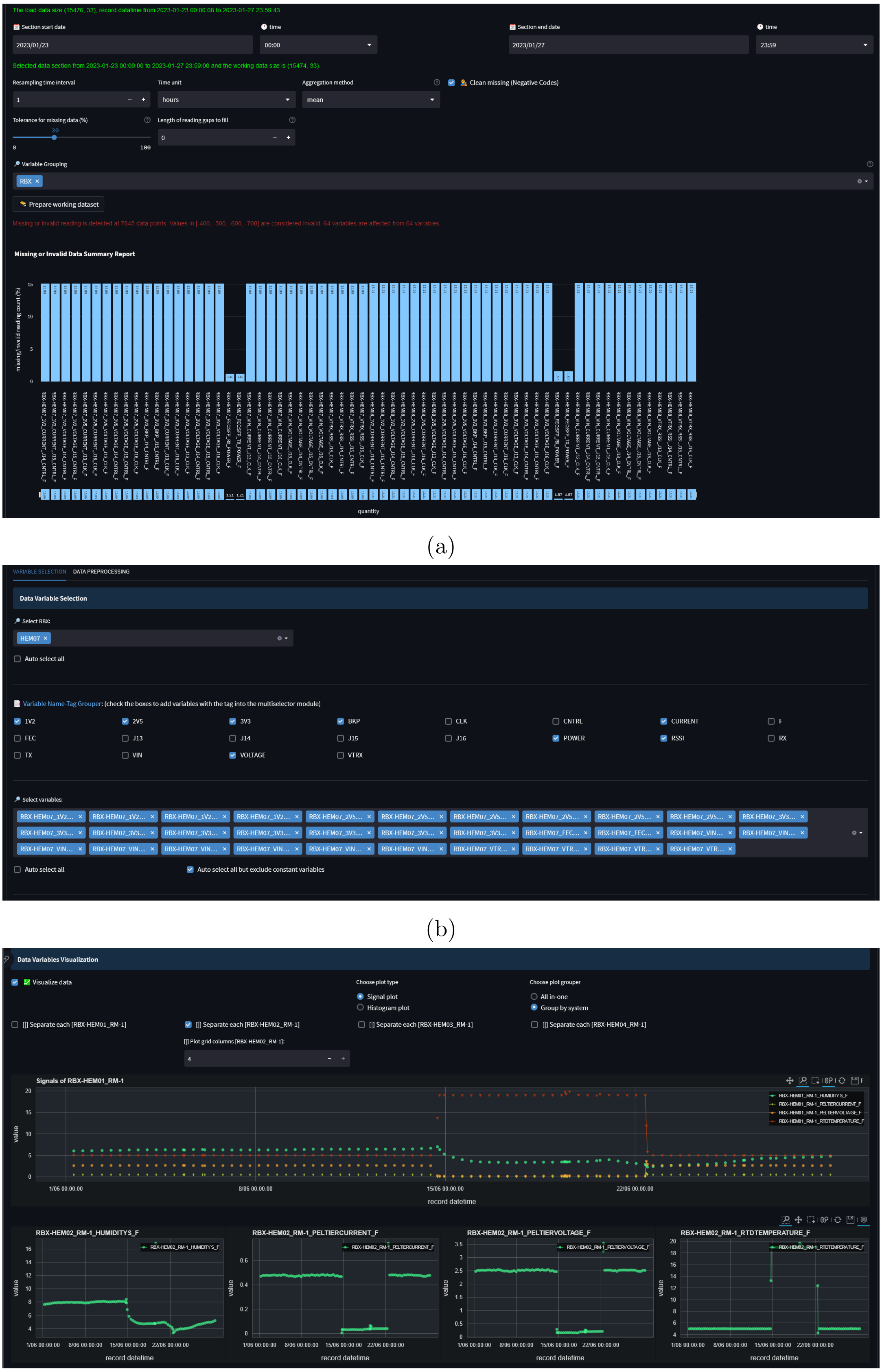
Length of reading gaps to fill: 0

Figure 7.2: The DESMOD dashboard sensor monitoring portal.

7.2 Data Quality Monitoring and Diagnostics Tools

The DQM AD page—shown in Fig. 7.10—provides access to AD inference using our spatio-temporal DQM-AD models for the HE and HB acquisition channels—discussed in Section 4.2 and 4.3. We render digi-occupancy map visualization and real bad channel detection results from our AD models at different system granularity, such as run-, lumisection-, and channel-granularity on recent Run-3 collision data (see from Fig. 7.11 to Fig. 7.14).

Fig. 7.11 portrays results of the spatio-temporal AD for HE three-dimensional DQM channel monitoring using our GraphSTAD deep learning model (see Section 4.2) [77]; the plots show user-setting selection, and summary AD reports at subdetector RBXes and channel granularity across the selected lumisections. Fig. 7.12 extends the illustration to lumisection granularity—single 3D map—showing the detected real faulty channels with details of the input, reconstructed, anomaly score spatial maps. We present similar results for the HB subdetector from Run-3 in Fig. 7.13 and Fig. 7.14 (see Section 4.2) [78].



The load data size (15476, 33), record datatime from 2023-01-23 00:00:08 to 2023-01-27 23:59:43

Section start date: 2023/01/23 — time: 00:00

Section end date: 2023/01/27 — time: 23:59

Selected data section from 2023-01-23 00:00:00 to 2023-01-27 23:59:00 and the working data size is (15474, 33).

Resampling time interval: 1

Time unit: hours

Aggregation method: mean

☑ Clean missing (Negative Codes)

Tolerance for missing data (%): 30 (0–100)

Length of reading gaps to fill: 0

Variable Grouping: RBX

Prepare working dataset

Missing or invalid reading is detected at 7845 data points. Values in [-400, -500, -600, -700] are considered invalid. 64 variables are affected from 64 variables.

Missing or Invalid Data Summary Report
missing/invalid reading count (%)
15
10
5
0
quantity

(a)

Data Variable Selection

Select RBX: HEM07

□ Auto select all

Variable Name-Tag Grouper: (check the boxes to add variables with the tag into the multiselector module)

☑ 1V2 ☑ 2V5 ☑ 3V3 ☑ BKP □ CLK □ CNTRL ☑ CURRENT □ F

□ FEC □ J13 □ J14 □ J15 □ J16 ☑ POWER ☑ RSSI □ RX

□ TX □ VIN ☑ VOLTAGE □ VTRX

Select variables:

RBX-HEM07_1V2... RBX-HEM07_1V2... RBX-HEM07_1V2... RBX-HEM07_1V2... RBX-HEM07_2V5... RBX-HEM07_2V5... RBX-HEM07_2V5... RBX-HEM07_2V5... RBX-HEM07_2V5... RBX-HEM07_2V5... RBX-HEM07_3V3...

RBX-HEM07_3V3... RBX-HEM07_3V3... RBX-HEM07_3V3... RBX-HEM07_3V3... RBX-HEM07_3V3... RBX-HEM07_3V3... RBX-HEM07_3V3... RBX-HEM07_FEC... RBX-HEM07_FEC... RBX-HEM07_VIN... RBX-HEM07_VIN...

RBX-HEM07_VIN... RBX-HEM07_VIN... RBX-HEM07_VIN... RBX-HEM07_VIN... RBX-HEM07_VIN... RBX-HEM07_VIN... RBX-HEM07_VTR... RBX-HEM07_VTR... RBX-HEM07_VTR... RBX-HEM07_VTR...

□ Auto select all ☑ Auto select all but exclude constant variables

(b)

Data Variables Visualization

☑ Visualize data

Choose plot type: ● Signal plot ○ Histogram plot

Choose plot grouper: ○ All in-one ● Group by system

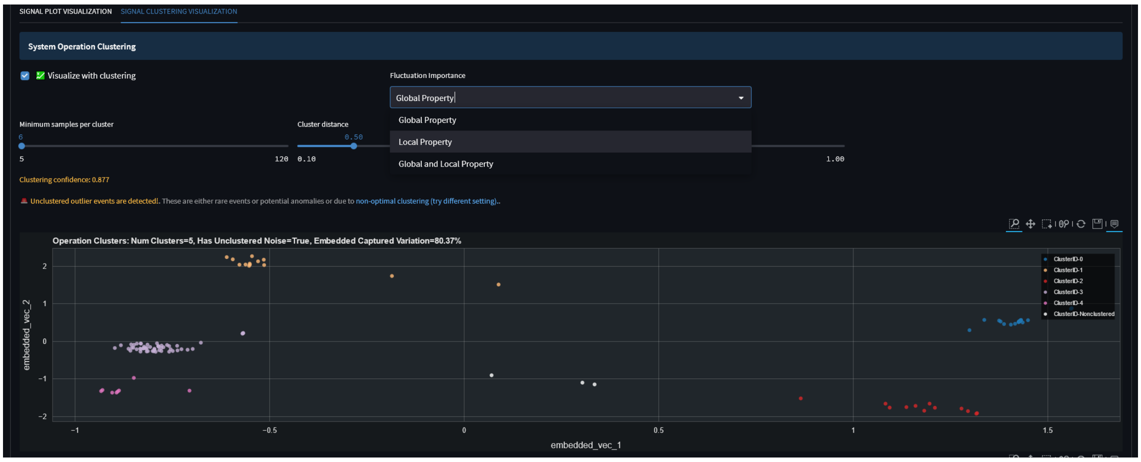
□ [|] Separate each [RBX-HEM01_RM-1] ☑ [|] Separate each [RBX-HEM02_RM-1] □ [|] Separate each [RBX-HEM03_RM-1] □ [|] Separate each [RBX-HEM04_RM-1]

[|] Plot grid columns [RBX-HEM02_RM-1]: 4

Signals of RBX-HEM01_RM-1
value
record datetime
1/06 00:00:00
8/06 00:00:00
15/06 00:00:00
22/06 00:00:00
RBX-HEM01_RM-1_HUMIDITYS_F
RBX-HEM01_RM-1_PELTIERCURRENT_F
RBX-HEM01_RM-1_PELTIERVOLTAGE_F
RBX-HEM01_RM-1_RTDTEMPERATURE_F
RBX-HEM02_RM-1_HUMIDITYS_F
RBX-HEM02_RM-1_PELTIERCURRENT_F
RBX-HEM02_RM-1_PELTIERVOLTAGE_F
RBX-HEM02_RM-1_RTDTEMPERATURE_F

(c)

Figure 7.3: The DESMOD dashboard multivariate TS data preprocessing panel: a) sensor data preprocessing and manipulation, b) system and sensor variable selection and filtering, and c) signal visualization.


SIGNAL PLOT VISUALIZATION
SIGNAL CLUSTERING VISUALIZATION

System Operation Clustering

☑ ✅ Visualize with clustering

Fluctuation Importance: Global Property

Global Property
Local Property
Global and Local Property

Minimum samples per cluster: 6 (5–120)

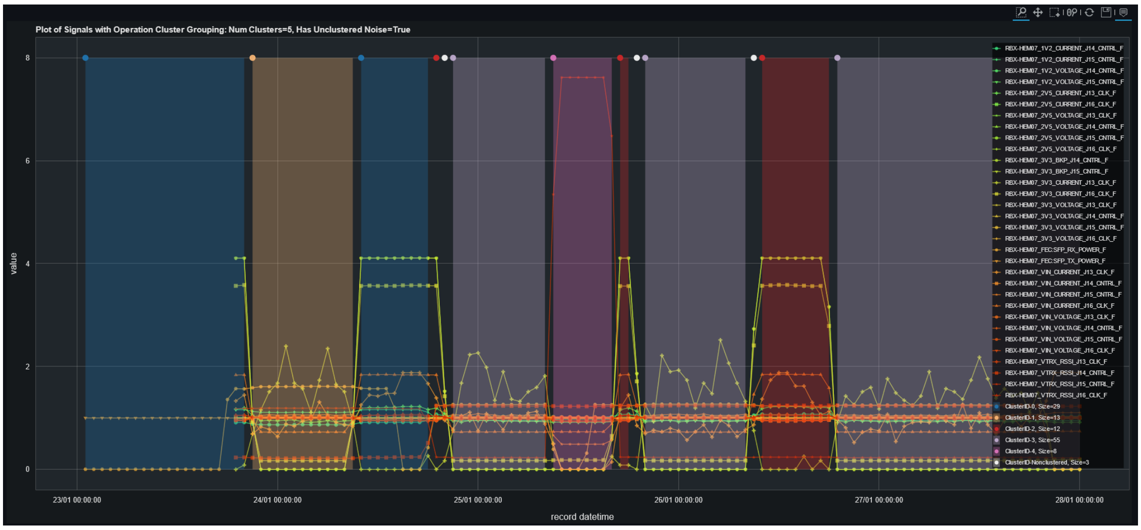
Cluster distance: 0.50 (0.10–1.00)

Clustering confidence: 0.877

Unclustered outlier events are detected!. These are either rare events or potential anomalies or due to non-optimal clustering (try different setting)..

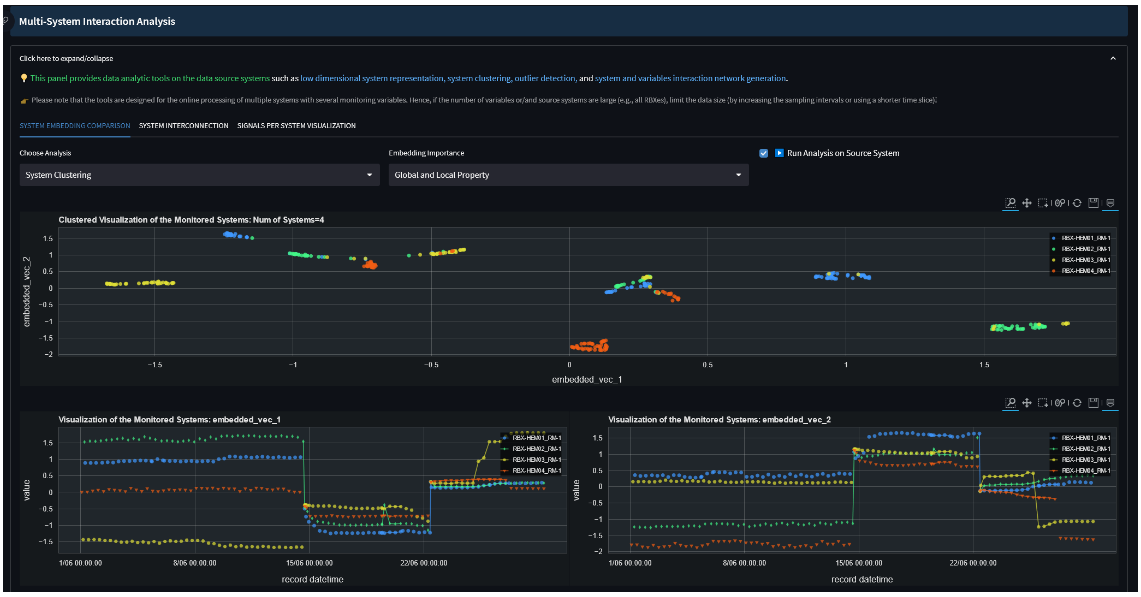
Operation Clusters: Num Clusters=5, Has Unclustered Noise=True, Embedded Captured Variation=80.37%
embedded_vec_2
2
1
0
−1
−2
−1
−0.5
0
0.5
1
1.5
embedded_vec_1
ClusterID-0
ClusterID-1
ClusterID-2
ClusterID-3
ClusterID-4
ClusterID-Nonclustered

(a)


Plot of Signals with Operation Cluster Grouping: Num Clusters=5, Has Unclustered Noise=True
value
8
6
4
2
0
23/01 00:00:00
24/01 00:00:00
25/01 00:00:00
26/01 00:00:00
27/01 00:00:00
28/01 00:00:00
record datetime
RBX-HEM07_1V2_CURRENT_J14_CNTRL_F
RBX-HEM07_1V2_CURRENT_J15_CNTRL_F
RBX-HEM07_1V2_VOLTAGE_J14_CNTRL_F
RBX-HEM07_1V2_VOLTAGE_J15_CNTRL_F
RBX-HEM07_2V5_CURRENT_J13_CLK_F
RBX-HEM07_2V5_CURRENT_J16_CLK_F
RBX-HEM07_2V5_VOLTAGE_J13_CLK_F
RBX-HEM07_2V5_VOLTAGE_J14_CNTRL_F
RBX-HEM07_2V5_VOLTAGE_J15_CNTRL_F
RBX-HEM07_2V5_VOLTAGE_J16_CLK_F
RBX-HEM07_3V3_BKP_J14_CNTRL_F
RBX-HEM07_3V3_BKP_J15_CNTRL_F
RBX-HEM07_3V3_CURRENT_J13_CLK_F
RBX-HEM07_3V3_CURRENT_J16_CLK_F
RBX-HEM07_3V3_VOLTAGE_J13_CLK_F
RBX-HEM07_3V3_VOLTAGE_J14_CNTRL_F
RBX-HEM07_3V3_VOLTAGE_J15_CNTRL_F
RBX-HEM07_3V3_VOLTAGE_J16_CLK_F
RBX-HEM07_FEC:SFP_RX_POWER_F
RBX-HEM07_FEC:SFP_TX_POWER_F
RBX-HEM07_VIN_CURRENT_J13_CLK_F
RBX-HEM07_VIN_CURRENT_J14_CNTRL_F
RBX-HEM07_VIN_CURRENT_J15_CNTRL_F
RBX-HEM07_VIN_CURRENT_J16_CLK_F
RBX-HEM07_VIN_VOLTAGE_J13_CLK_F
RBX-HEM07_VIN_VOLTAGE_J14_CNTRL_F
RBX-HEM07_VIN_VOLTAGE_J15_CNTRL_F
RBX-HEM07_VIN_VOLTAGE_J16_CLK_F
RBX-HEM07_VTRX_RSSI_J13_CLK_F
RBX-HEM07_VTRX_RSSI_J14_CNTRL_F
RBX-HEM07_VTRX_RSSI_J15_CNTRL_F
RBX-HEM07_VTRX_RSSI_J16_CLK_F
ClusterID-0, Size=29
ClusterID-1, Size=13
ClusterID-2, Size=12
ClusterID-3, Size=55
ClusterID-4, Size=8
ClusterID-Nonclustered, Size=3

(b)

Figure 7.4: The DESMOD dashboard multivariate TS data clustering: a) annotated scatter plot, and b) annotated signal plot.



Multi-System Interaction Analysis

Click here to expand/collapse

💡 This panel provides data analytic tools on the data source systems such as low dimensional system representation, system clustering, outlier detection, and system and variables interaction network generation.

👉 Please note that the tools are designed for the online processing of multiple systems with several monitoring variables. Hence, if the number of variables or/and source systems are large (e.g., all RBXes), limit the data size (by increasing the sampling intervals or using a shorter time slice)!

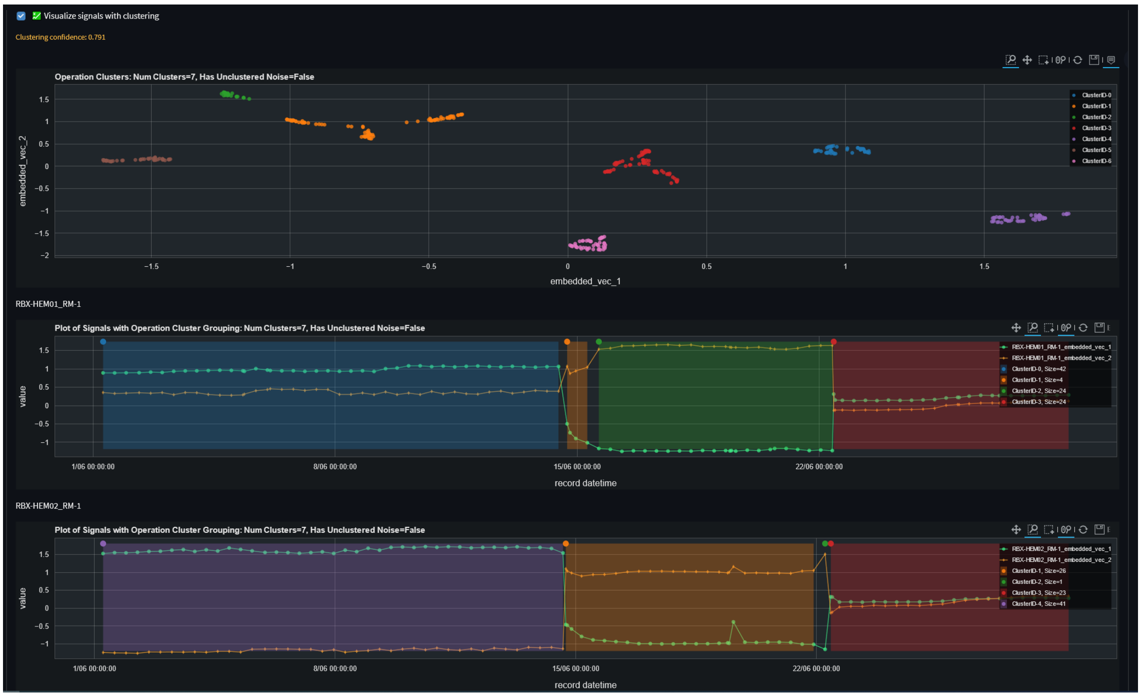
SYSTEM EMBEDDING COMPARISON | SYSTEM INTERCONNECTION | SIGNALS PER SYSTEM VISUALIZATION

Choose Analysis: System Clustering

Embedding Importance: Global and Local Property

☑ ▶ Run Analysis on Source System

(a)

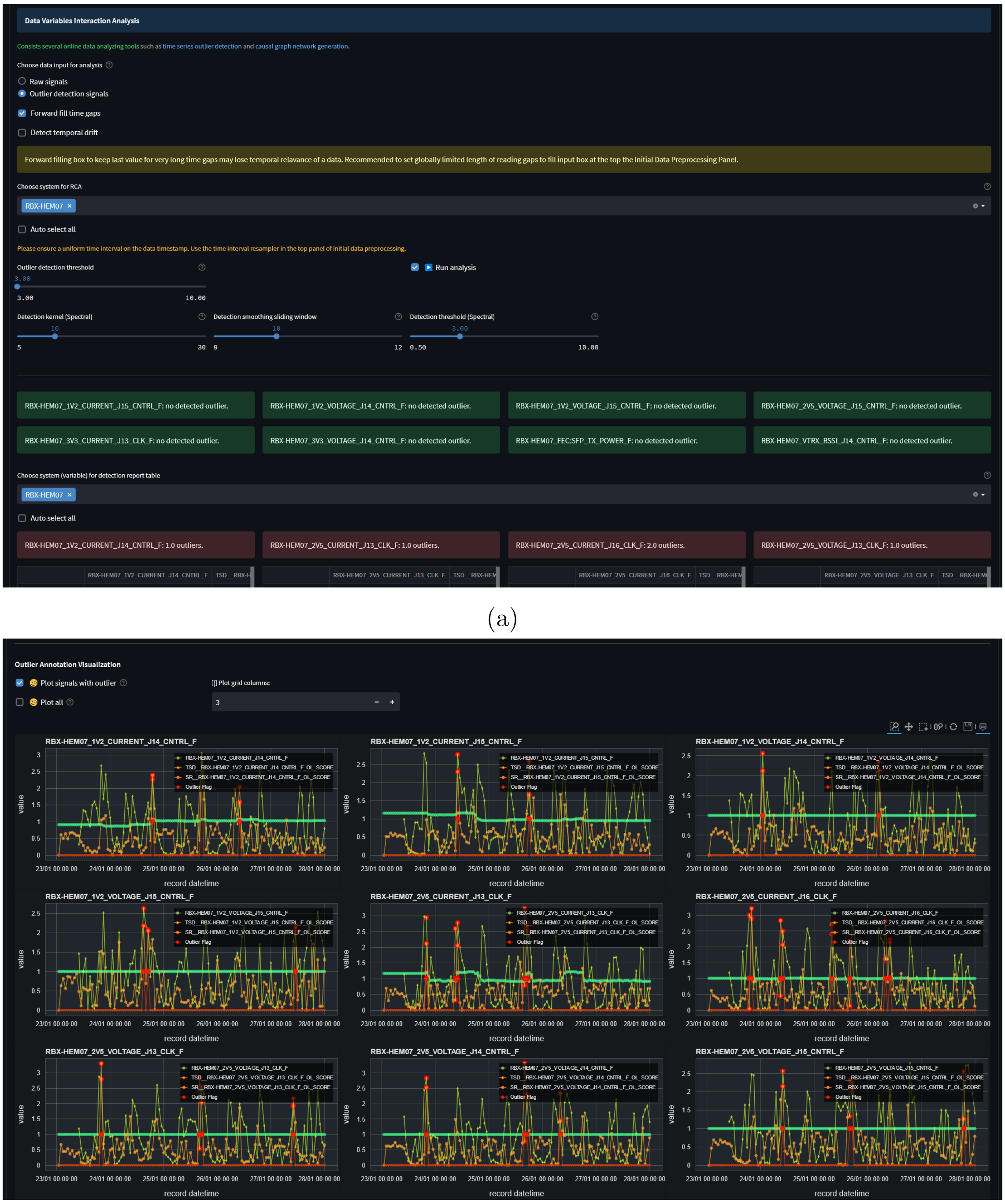


☑ ✅ Visualize signals with clustering

Clustering confidence: 0.791

(b)

Figure 7.5: The DESMOD dashboard RBX system clustering based on dimensional reduction embedding on multivariate sensors: a) 2-dimensional embedding of multiple systems, and b) system deviation detection through signal clustering.



Data Variables Interaction Analysis

Consists several online data analyzing tools such as time series outlier detection and causal graph network generation.

Choose data input for analysis

○ Raw signals
● Outlier detection signals

☑ Forward fill time gaps

☐ Detect temporal drift

Forward filling box to keep last value for very long time gaps may lose temporal relavance of a data. Recommended to set globally limited length of reading gaps to fill input box at the top the Initial Data Preprocessing Panel.

Choose system for RCA

RBX-HEM07 ×

☐ Auto select all

Please ensure a uniform time interval on the data timestamp. Use the time interval resampler in the top panel of initial data preprocessing.

Outlier detection threshold: 3.00 (3.00 – 10.00)

☑ Run analysis

Detection kernel (Spectral): 10 (5 – 30)

Detection smoothing sliding window: 10 (9 – 12)

Detection threshold (Spectral): 3.00 (0.50 – 10.00)

RBX-HEM07_1V2_CURRENT_J15_CNTRL_F: no detected outlier.

RBX-HEM07_1V2_VOLTAGE_J14_CNTRL_F: no detected outlier.

RBX-HEM07_1V2_VOLTAGE_J15_CNTRL_F: no detected outlier.

RBX-HEM07_2V5_VOLTAGE_J15_CNTRL_F: no detected outlier.

RBX-HEM07_3V3_CURRENT_J13_CLK_F: no detected outlier.

RBX-HEM07_3V3_VOLTAGE_J14_CNTRL_F: no detected outlier.

RBX-HEM07_FEC:SFP_TX_POWER_F: no detected outlier.

RBX-HEM07_VTRX_RSSI_J14_CNTRL_F: no detected outlier.

Choose system (variable) for detection report table

RBX-HEM07 ×

☐ Auto select all

RBX-HEM07_1V2_CURRENT_J14_CNTRL_F: 1.0 outliers.

RBX-HEM07_2V5_CURRENT_J13_CLK_F: 1.0 outliers.

RBX-HEM07_2V5_CURRENT_J16_CLK_F: 2.0 outliers.

RBX-HEM07_2V5_VOLTAGE_J13_CLK_F: 1.0 outliers.

	RBX-HEM07_1V2_CURRENT_J14_CNTRL_F	TSD__RBX-H

	RBX-HEM07_2V5_CURRENT_J13_CLK_F	TSD__RBX-HEM

	RBX-HEM07_2V5_CURRENT_J16_CLK_F	TSD__RBX-HEM

	RBX-HEM07_2V5_VOLTAGE_J13_CLK_F	TSD__RBX-HEM

(a)

Outlier Annotation Visualization

☑ 🧐 Plot signals with outlier

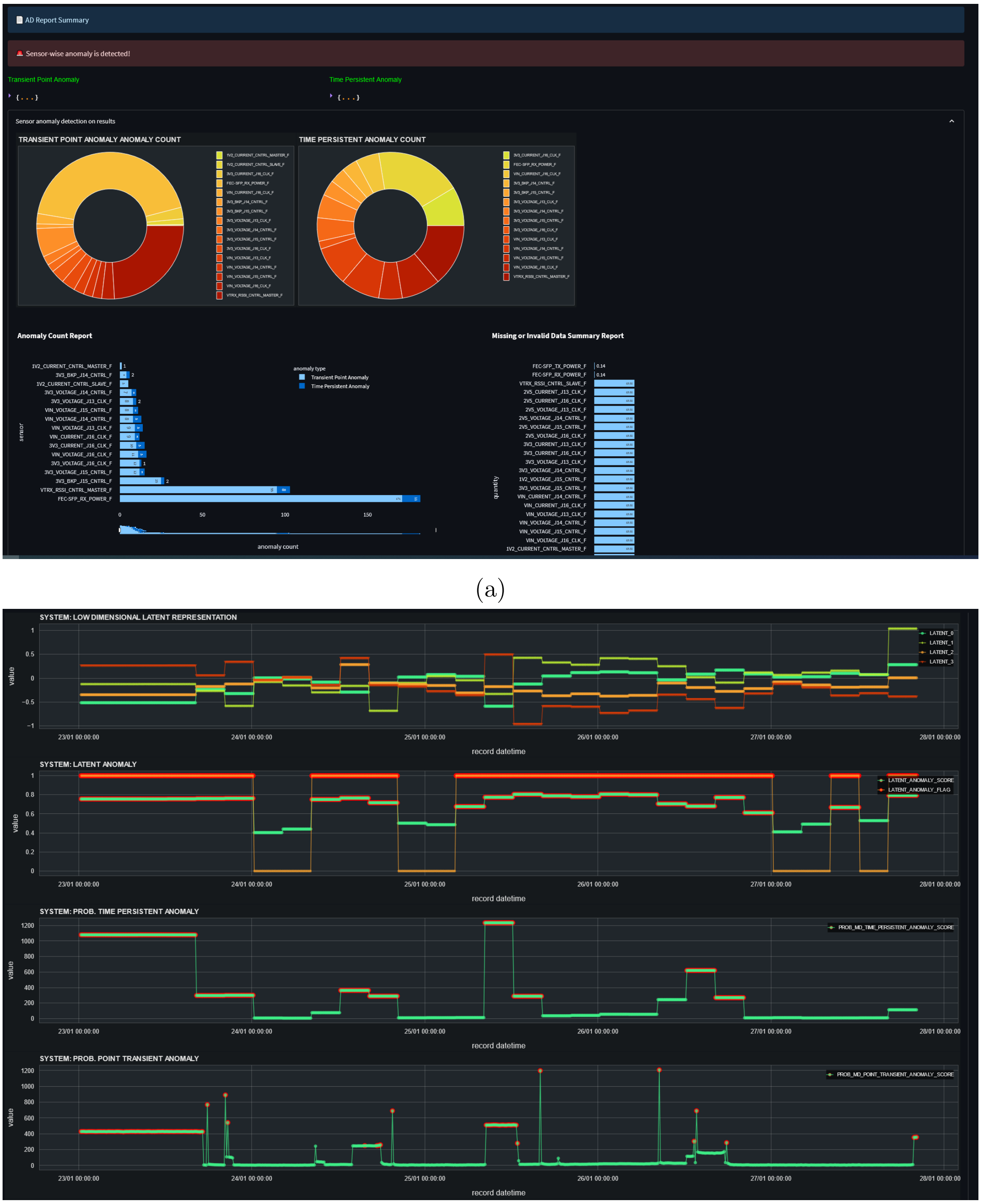
☐ 🧐 Plot all

[|] Plot grid columns: 3

RBX-HEM07_1V2_CURRENT_J14_CNTRL_F
RBX-HEM07_1V2_CURRENT_J15_CNTRL_F
RBX-HEM07_1V2_VOLTAGE_J14_CNTRL_F
RBX-HEM07_1V2_VOLTAGE_J15_CNTRL_F
RBX-HEM07_2V5_CURRENT_J13_CLK_F
RBX-HEM07_2V5_CURRENT_J16_CLK_F
RBX-HEM07_2V5_VOLTAGE_J13_CLK_F
RBX-HEM07_2V5_VOLTAGE_J14_CNTRL_F
RBX-HEM07_2V5_VOLTAGE_J15_CNTRL_F
TSD__RBX-HEM07_1V2_CURRENT_J14_CNTRL_F_OL_SCORE
SR__RBX-HEM07_1V2_CURRENT_J14_CNTRL_F_OL_SCORE
Outlier Flag
value
record datetime
23/01 00:00:00
24/01 00:00:00
25/01 00:00:00
26/01 00:00:00
27/01 00:00:00
28/01 00:00:00
0
0.5
1
1.5
2
2.5
3

(b)

Figure 7.6: The DESMOD dashboard online AD without a trained model: a) AD configuration setting panel and AD summary report, and b) AD result plots.



AD Report Summary

Sensor-wise anomaly is detected!

Transient Point Anomaly

{...}

Time Persistent Anomaly

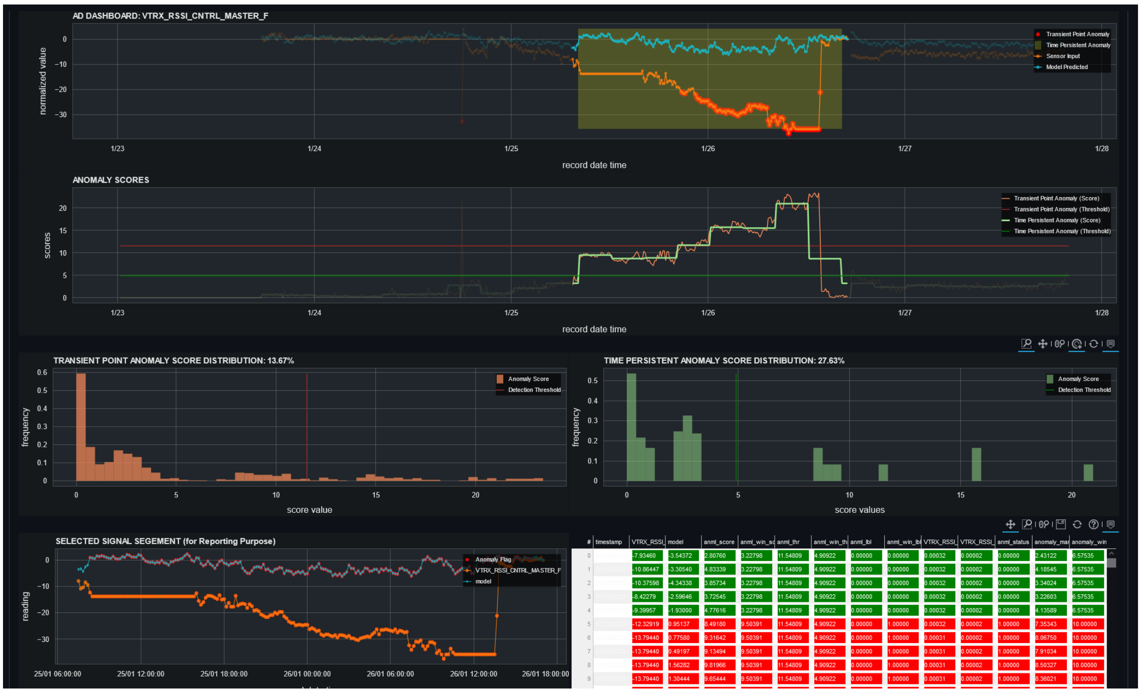
{...}

Sensor anomaly detection on results

(a)

(b)

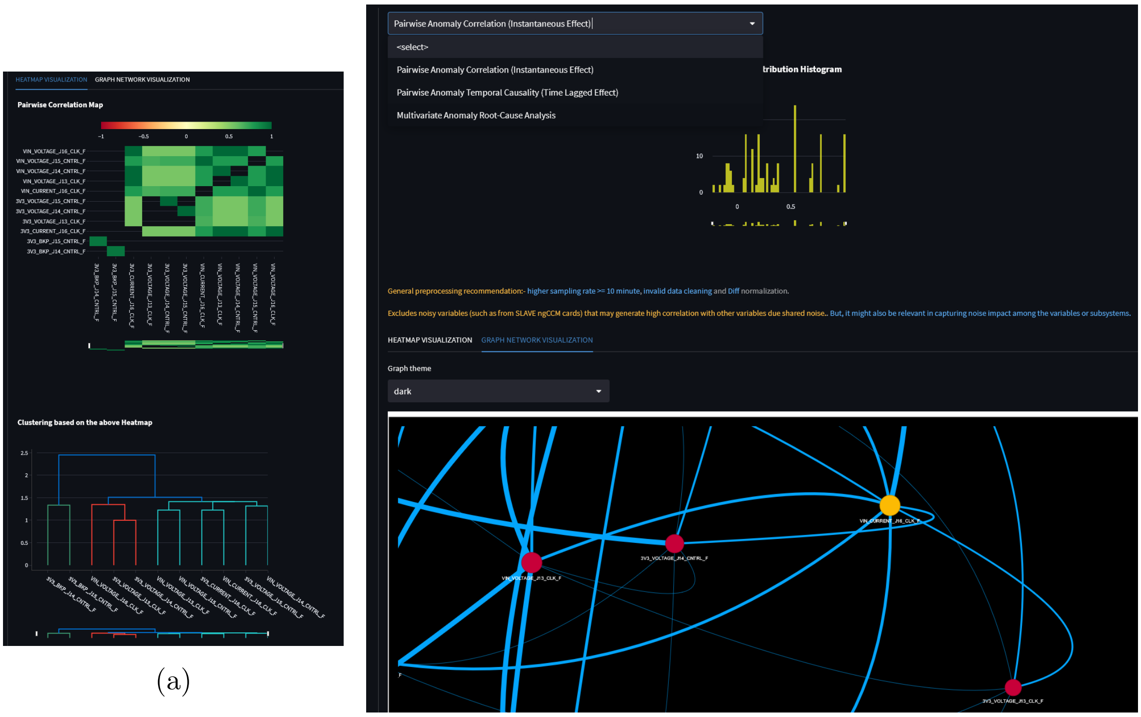
Figure 7.7: The DESMOD dashboard HEngCCM AD prediction using trained deep learning CGVAE model (see Section 4.1): a) sensor-level AD report summary, and b) system-level AD report.



#	timestamp	VTRX_RSSI_	model	anml_score	anml_win_sc	anml_thr	anml_win_th	anml_lbl	anml_win_lbl	VTRX_RSSI_	VTRX_RSSI_	anml_status	anomaly_mar	anomaly_win
0	01/25/2023 0	-7.93460	-3.54372	2.80760	3.22798	11.54809	4.90922	0.00000	0.00000	0.00032	0.00002	0.00000	2.43122	6.57535
1	01/25/2023 0	-10.86447	-3.30540	4.83339	3.22798	11.54809	4.90922	0.00000	0.00000	0.00032	0.00002	0.00000	4.18545	6.57535
2	01/25/2023 0	-10.37598	-4.34338	3.85734	3.22798	11.54809	4.90922	0.00000	0.00000	0.00032	0.00002	0.00000	3.34024	6.57535
3	01/25/2023 0	-8.42279	-2.59646	3.72545	3.22798	11.54809	4.90922	0.00000	0.00000	0.00032	0.00002	0.00000	3.22603	6.57535
4	01/25/2023 0	-9.39957	-1.93000	4.77616	3.22798	11.54809	4.90922	0.00000	0.00000	0.00032	0.00002	0.00000	4.13589	6.57535
5	01/25/2023 0	-12.32919	0.95137	8.49180	9.50391	11.54809	4.90922	0.00000	1.00000	0.00032	0.00002	1.00000	7.35343	10.00000
6	01/25/2023 0	-13.79440	0.77580	9.31642	9.50391	11.54809	4.90922	0.00000	1.00000	0.00031	0.00002	1.00000	8.06750	10.00000
7	01/25/2023 0	-13.79440	0.49197	9.13494	9.50391	11.54809	4.90922	0.00000	1.00000	0.00031	0.00002	1.00000	7.91034	10.00000
8	01/25/2023 0	-13.79440	1.56282	9.81966	9.50391	11.54809	4.90922	0.00000	1.00000	0.00031	0.00002	1.00000	8.50327	10.00000
9	01/25/2023 0	-13.79440	1.30444	9.65444	9.50391	11.54809	4.90922	0.00000	1.00000	0.00031	0.00002	1.00000	8.36021	10.00000

(a)

Figure 7.8: The DESMOD dashboard HEngCCM AD using trained CGVAE model (see Section 4.1): AD report on a selected RSSI sensor with gradual drifting anomaly.



General preprocessing recommendation:- higher sampling rate >= 10 minute, invalid data cleaning and Diff normalization.

Excludes noisy variables (such as from SLAVE ngCCM cards) that may generate high correlation with other variables due shared noise.. But, it might also be relevant in capturing noise impact among the variables or subsystems.

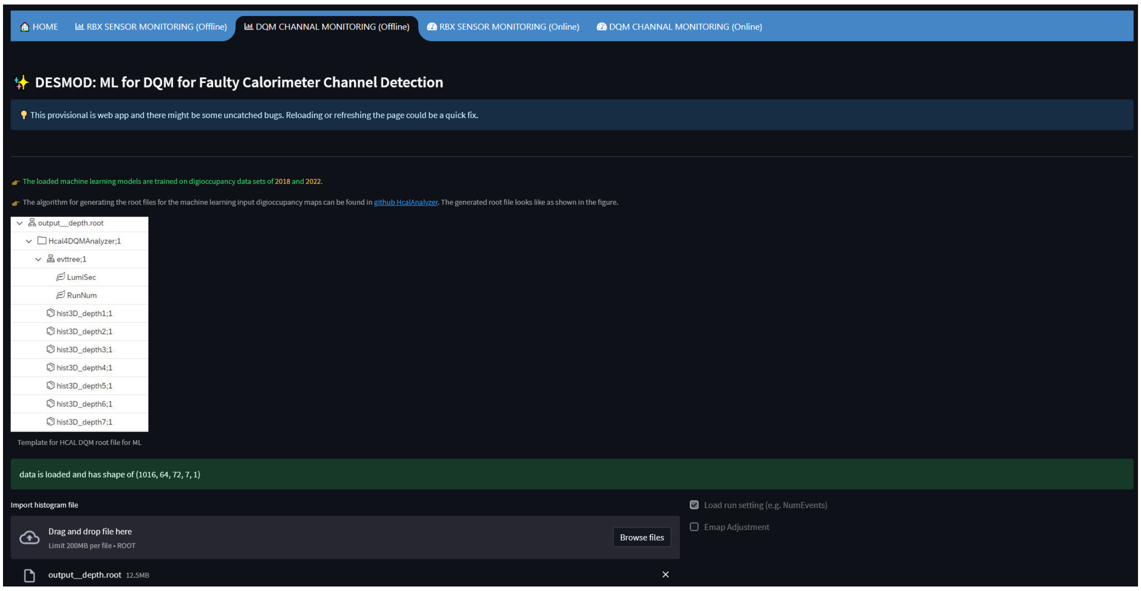
Graph theme

dark

(a)

(b)

Figure 7.9: The DESMOD dashboard sensor interconnection and causality analysis on reported anomaly flag data (see Section 6.2): a) similarity heatmap and clustering dendrogram, and b) interconnection graph network.


HOME
RBX SENSOR MONITORING (Offline)
DQM CHANNAL MONITORING (Offline)
RBX SENSOR MONITORING (Online)
DQM CHANNAL MONITORING (Online)

✨ DESMOD: ML for DQM for Faulty Calorimeter Channel Detection

💡 This provisional is web app and there might be some uncatched bugs. Reloading or refreshing the page could be a quick fix.

👉 The loaded machine learning models are trained on digioccupancy data sets of 2018 and 2022.

👉 The algorithm for generating the root files for the machine learning input digioccupancy maps can be found in github HcalAnalyzer. The generated root file looks like as shown in the figure.

```
output__depth.root
  Hcal4DQMAnalyzer;1
    evttree;1
      LumiSec
      RunNum
    hist3D_depth1;1
    hist3D_depth2;1
    hist3D_depth3;1
    hist3D_depth4;1
    hist3D_depth5;1
    hist3D_depth6;1
    hist3D_depth7;1
```

Template for HCAL DQM root file for ML

data is loaded and has shape of (1016, 64, 72, 7, 1)

Import histogram file

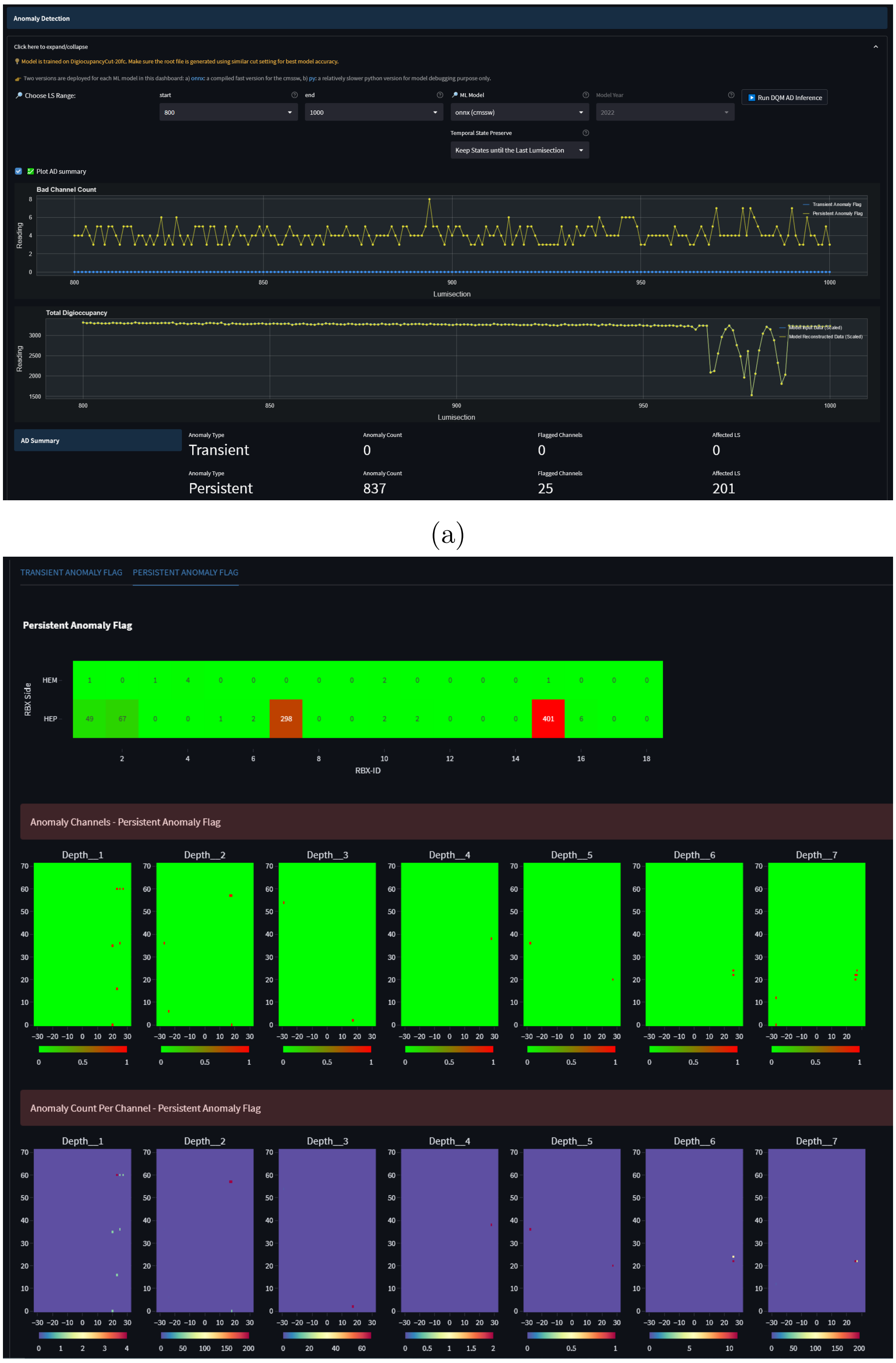
Drag and drop file here
Limit 200MB per file • ROOT

☑ Load run setting (e.g. NumEvents)

☐ Emap Adjustment

output__depth.root 12.5MB

Figure 7.10: The DESMOD dashboard anomaly detection for the HCAL data quality monitoring.



Anomaly Detection

Click here to expand/collapse

Model is trained on DigiocupancyCut-20fc. Make sure the root file is generated using similar cut setting for best model accuracy.

Two versions are deployed for each ML model in this dashboard: a) onnx: a compiled fast version for the cmssw, b) py: a relatively slower python version for model debugging purpose only.

Choose LS Range: start 800, end 1000

ML Model: onnx (cmssw)

Model Year: 2022

Temporal State Preserve: Keep States until the Last Lumisection

☑ Plot AD summary

AD Summary	Anomaly Type	Anomaly Count	Flagged Channels	Affected LS
	Transient	0	0	0
	Persistent	837	25	201

(a)

TRANSIENT ANOMALY FLAG

PERSISTENT ANOMALY FLAG

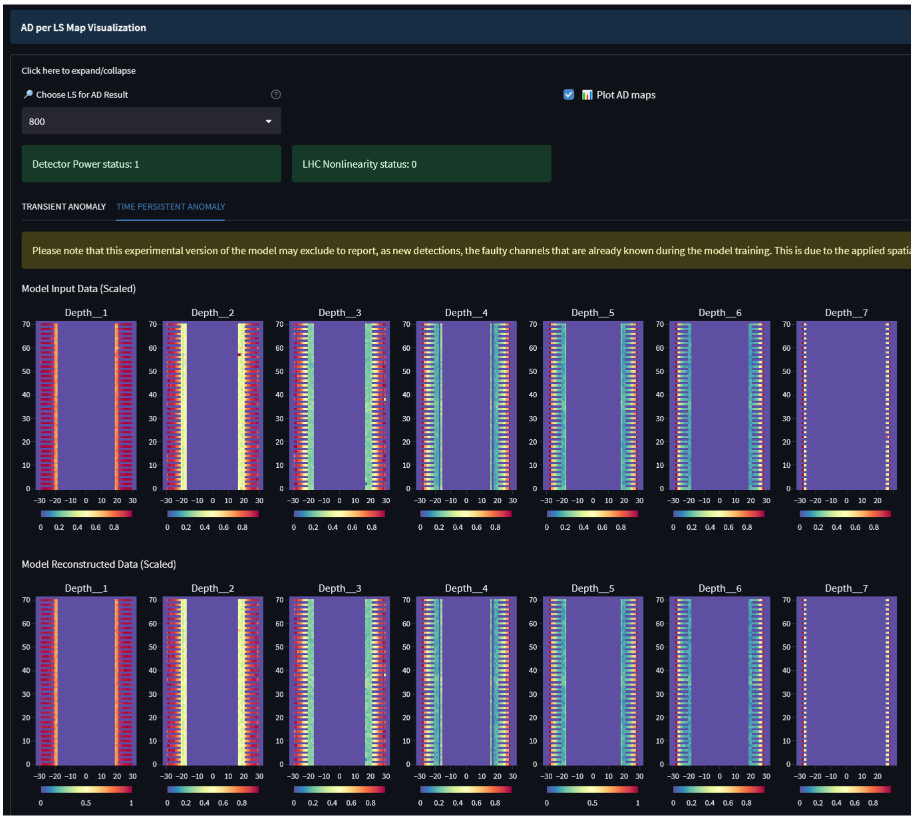
Persistent Anomaly Flag

Anomaly Channels - Persistent Anomaly Flag

Anomaly Count Per Channel - Persistent Anomaly Flag

(b)

Figure 7.11: The DESMOD dashboard GraphSTAD for HE-DQM channel monitoring on $LS \in [800, 1000]$ of the *RunId=361240* data (see Section 4.2): a) inference user-settings and AD summary report across lumisections, and b) AD summary report per channel hosting RBXes on the 3D spatial DQM map.



AD per LS Map Visualization

Click here to expand/collapse

Choose LS for AD Result

800

☑ Plot AD maps

Detector Power status: 1

LHC Nonlinearity status: 0

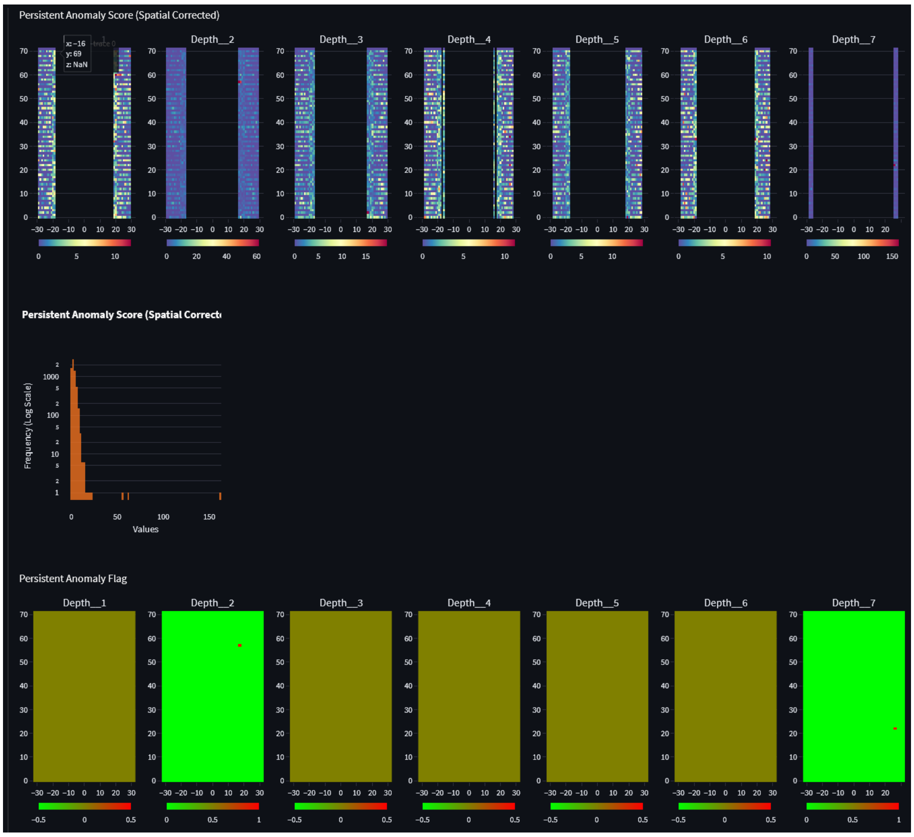
TRANSIENT ANOMALY TIME PERSISTENT ANOMALY

Please note that this experimental version of the model may exclude to report, as new detections, the faulty channels that are already known during the model training. This is due to the applied spati

Model Input Data (Scaled)

Model Reconstructed Data (Scaled)

(a)



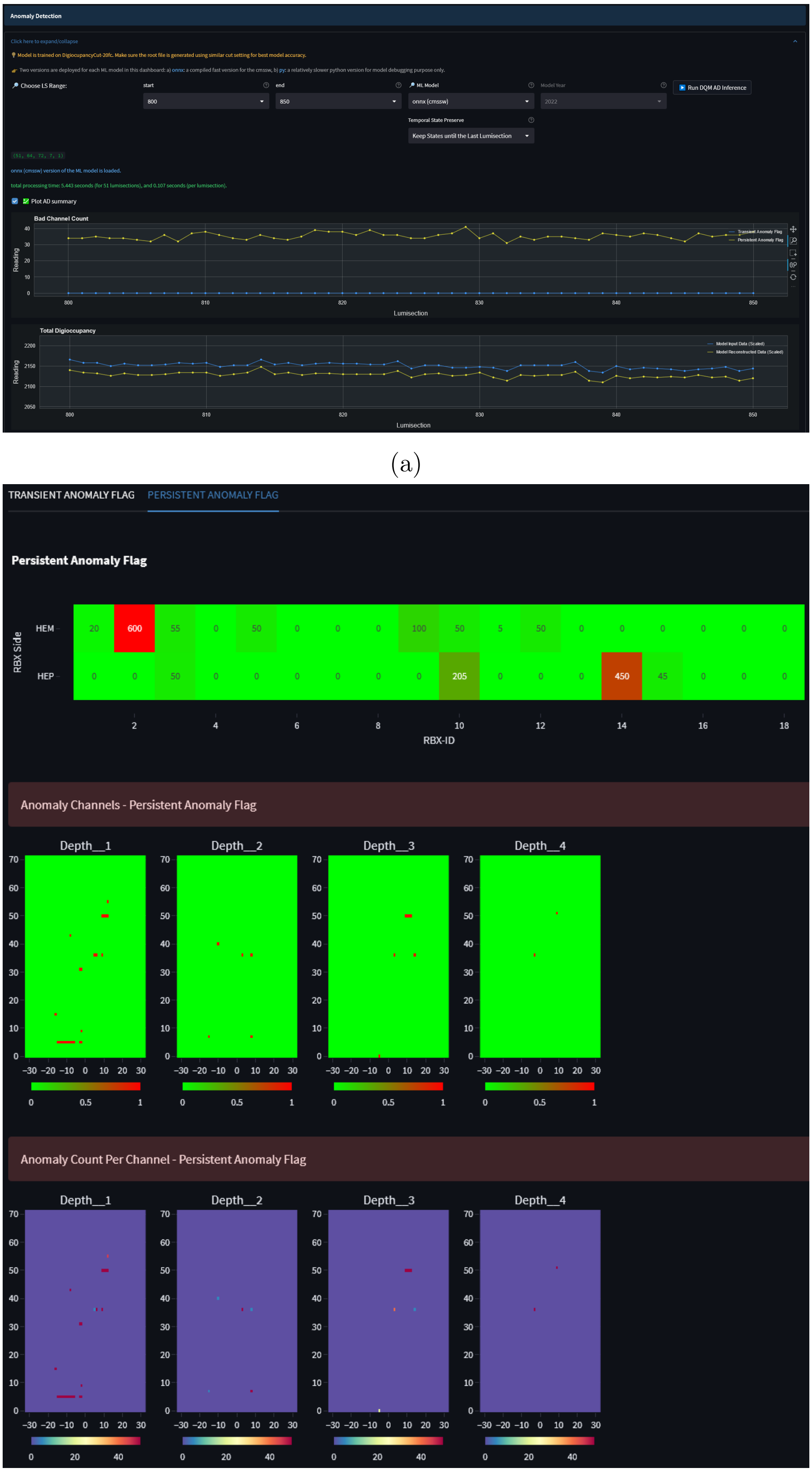
Persistent Anomaly Score (Spatial Corrected)

Persistent Anomaly Score (Spatial Correct

Persistent Anomaly Flag

(b)

Figure 7.12: The DESMOD dashboard GraphSTAD report on a selected HE-DQM map at *LS=800* from *RunId=361240*: a) the input and predicted maps, and b) anomaly score and flag report.



Anomaly Detection

Click here to expand/collapse

💡 Model is trained on DigiocupancyCut-20fc. Make sure the root file is generated using similar cut setting for best model accuracy.

👉 Two versions are deployed for each ML model in this dashboard: a) onnx: a compiled fast version for the cmssw, b) py: a relatively slower python version for model debugging purpose only.

🔎 Choose LS Range: start 800, end 850

🔎 ML Model: onnx (cmssw)

Model Year: 2022

Run DQM AD Inference

Temporal State Preserve: Keep States until the Last Lumisection

```
(51, 64, 72, 7, 1)
```

onnx (cmssw) version of the ML model is loaded.

total processing time: 5.443 seconds (for 51 lumisections), and 0.107 seconds (per lumisection).

☑ ✅ Plot AD summary

(a)

TRANSIENT ANOMALY FLAG PERSISTENT ANOMALY FLAG

Persistent Anomaly Flag

RBX Side		2		4		6		8		10		12		14		16		18
HEM	20	600	55	0	50	0	0	0	100	50	5	50	0	0	0	0	0	0
HEP	0	0	50	0	0	0	0	0	0	205	0	0	0	450	45	0	0	0

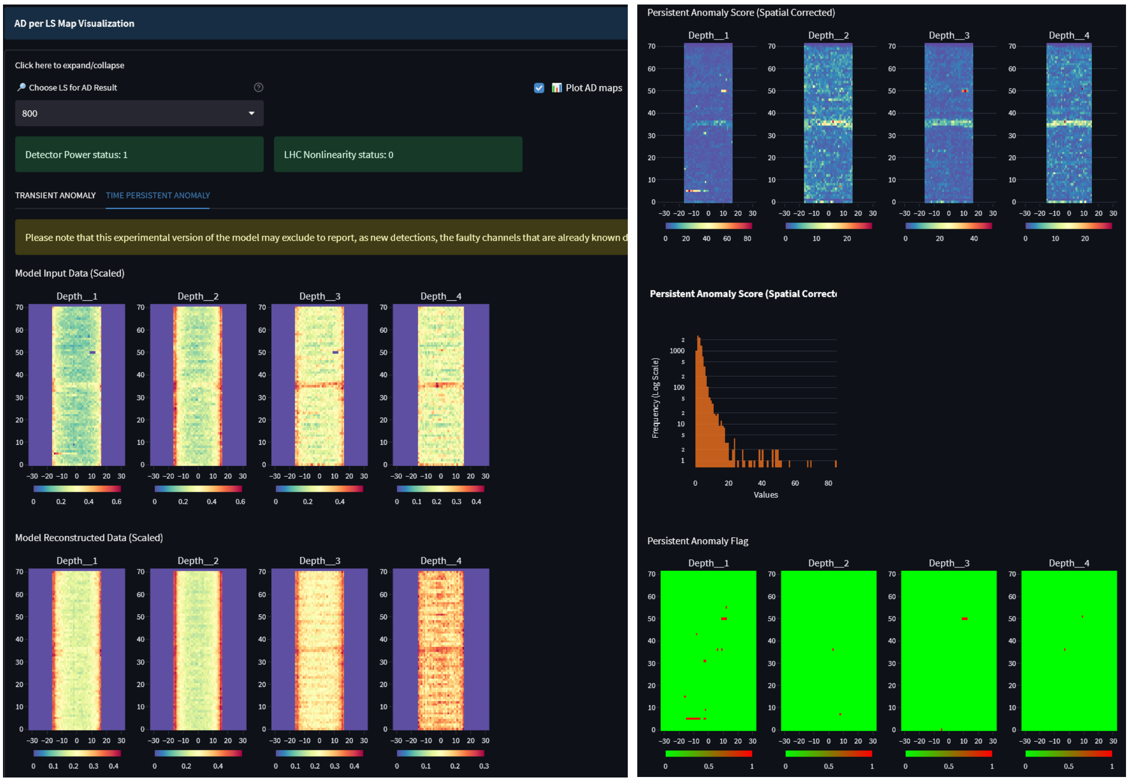
RBX-ID

Anomaly Channels - Persistent Anomaly Flag

Anomaly Count Per Channel - Persistent Anomaly Flag

(b)

Figure 7.13: The DESMOD dashboard GraphSTAD for HB-DQM channel monitoring on $LS \in [800, 850]$ of the *RunId=361240* data (see Section 4.2): a) inference user-settings and AD summary report across lumisections, and b) AD summary report per channel hosting RBXes on the 3D spatial DQM map.



AD per LS Map Visualization

Click here to expand/collapse

Choose LS for AD Result: 800

☑ Plot AD maps

Detector Power status: 1

LHC Nonlinearity status: 0

TRANSIENT ANOMALY | TIME PERSISTENT ANOMALY

Please note that this experimental version of the model may exclude to report, as new detections, the faulty channels that are already known d

Model Input Data (Scaled)

Model Reconstructed Data (Scaled)

Persistent Anomaly Score (Spatial Corrected)

Persistent Anomaly Score (Spatial Correct

Persistent Anomaly Flag

(a)

(b)

Figure 7.14: The DESMOD dashboard GraphSTAD report on a selected HB-DQM map at *LS=800* from *RunId=361240*: a) the input and predicted maps, and b) anomaly score and flag report.

Chapter 8

Conclusions

This chapter will summarize the scientific contribution and research impact of our study, and state some potential future research questions.

8.1 Scientific Contributions

The author of this dissertation has made significant contributions to this study—including data curation and preparation, methodology development, implementation and experimentation, and preparation and review of the manuscripts. The HCAL experts have participated in data collection, technical discussions, result validation, and manuscript reviews. We will highlight below our scientific contribution to the research questions outlined in Section 1.4. The Ukraine war sanctions delay the manuscripts marked with (*).

Our study accomplishes monitoring of complex systems of the HCAL detector through a divide-and-conquer approach. We break down the monitoring of the detector systems into smaller subsystems. We exploit behavioral similarities among the subsystems and handle differences—primarily through normalization techniques and deep learning models—to develop generic models per subsystem groups.

We have developed the CGVAE model, a data-driven unsupervised AD using deep learning, for multivariate time series sensor data when addressing RQ1 (see Chapter 4 and **Paper-1** in Ref. [41]). The CGVAE model combines encoded latent feature- and reconstruction-based metrics for enhanced AD to mitigate signal reconstruction overfitting on anomalous patterns. It integrates feature attribution algorithms for TS data to explain the contribution of the input sensors to the detected anomalies.

We have proposed predicting anomalies through DL models from early warning symptoms on multivariate sensor data when addressing RQ2 (see Chapter 5 and **Paper-2** in Ref. [70]). We have introduced AnoP, a long horizon multi-timestep anomaly prediction system using causal residual networks forecasting and AD DL models, to raise alerts for anomaly prevention. The forecasting model presents an attention-based multi-horizon prediction S2S model.

We have developed GraphSTAD, which integrates spatial and temporal learning networks, for AD monitoring on high-dimensional spatio-temporal data when addressing RQ4 (see Chapter 4, **Paper-3** in Ref. [76] and **Paper-4** in Ref. [77]). GraphSTAD employs regular and irregular spatial characteristics learning mechanisms using CNNs and GNNs to capture context and substantially enhance AD performance.

We have investigated transfer learning on complex spatio-temporal AD DL networks when addressing RQ5. We have studied both the potential benefits and limitations of different configurations of transfer learning on a multi-network DL model (CNNs, GNNs, and LSTMs) within the context of spatio-temporal AD between the HCAL HE and HB subdetector systems (see Chapter 4 and **Paper-5*** in Ref. [78]).

We have devised a lightweight multivariate analysis approach for online interconnection divergence discovery in multi-systems environment, and a computationally efficient AnomalyCD approach for discovering causal graphs on multivariate binary anomaly data when addressing RQ3 (see Chapter 6, **Paper-6** in Ref. [73] and **Paper-7*** in Ref. [74]). The interconnection divergence discovery approach offers a simple online mechanism—without heavy data preprocessing, data annotation, and pre-training requirements—for fast discovery of outlier behaviors through hierarchical interconnection clustering analysis on multi-systems with multivariate sensors. The AnomalyCD framework consists of multiple strategies to overcome the challenges of generating anomaly causality graphs through unsupervised online anomaly detection, sparse data and link handling, and edge adjustment approaches.

8.2 Research Impact

Our data-driven monitoring tools have achieved production accuracy and are deployed at CERN. The tools have expanded the monitoring capability of the Hadron Calorimeter of the CMS experiment—capturing previously unknown and challenging front-end electronics anomalies. We have integrated the data quality monitoring AD models into the CMSSW core production system for online and offline monitoring of the collision experiments. We have also developed a DESMOD dashboard, which is deployed at CERN, to host the infrastructure AD models. We will highlight below the research impact of our study.

- *Infrastructure monitoring*: The CGVAE anomaly detection and AnoP anomaly prediction models, discussed in Section 4.1 and 5.1, respectively, monitor approximately 1K diagnostics sensors of the ngCCM system of the thirty-six readout boxes of the HE subdetector. The tools have broadened anomaly monitoring automation and detected previously unseen and challenging-to-capture anomalies, such as collective slowly drifting anomalies. Our tools have discovered two previously strange anomalies in the calorimeter's sensors: 1) our models have detected signal degradation on the low-voltage supply of the detector unexpectedly caused by the machine development and technical stop tasks at the LHC; this new knowledge allows the HCAL operation team to better prepare for LHC interventions in the future, and 2) we have detected noise anomaly on the few slave control cards of the ngCCM system due to the FPGA modules of the cards attempt to lock onto a non-existent incoming data stream; it is not unexpected for the slave card to behave in this manner, but monitoring its status would provide relevant information when making master-slave configuration switching. We have deployed the tools in the DESMOD dashboard to provide sensor monitoring and diagnostic services to the HCAL experts.

- *Data quality monitoring*: We have successfully integrated our GraphSTAD models for high dimensional DQM AD, presented in Section 4.2 and 4.3, in the CMSSW core production. The tools have become part of the online DQM to actively monitor about 7K and 9K particle acquisition channels of the HE and HB subdetectors, respectively. The models have detected and accurately localized real faulty channels—dead, hot, and degrading—with 23-second lumisection granularity.

- *Anomaly Causality Discovery*: Our approaches proposed in Chapter 6 have enabled quick and computationally efficient interaction and discrepancy discovery among RBX systems monitors by multivariate sensors. The AnomalyCD provides causality knowledge and online inference on binary anomaly data streamed from different systems of the HCAL; Our online tools have monitored approximately 2K and 16K sensors of the 4 RMs of the 36 RBXes of the HE from the HCAL MonDB database with 12 sensors per RM and the ngCCM server with 113 sensors per RM, respectively. We have incorporated the tools in the DESMOD dashboard.

8.3 Future Research Directions

Our research raises the following potential research questions.

- Extending the machine learning automation effort at the HCAL: Our study has primarily discussed the endcap and barrel detectors of the HCAL. Extending the ML efforts into the other subdetectors that are not covered in our study—the outer and forward detectors of the HCAL—would be an attractive research direction to pursue, having the encouraging results of our study.

- Robust data normalization toward generic deep learning models for TSAD: The lack of a robust generic normalization method has deteriorated performance in several DL TSAD models [12,81,130]. Machine learning models may experience data concept drift during inference after training [348]. Deep learning models may struggle to maintain accuracy even when trained on normalized data sets. Recent studies in Refs. [41,70,185,348] highlight the importance of normalization in deep learning for time series modeling. A seasonal-trend decomposition significantly leverages transformers for TS forecasting [185]. Reversible instance normalization standardizes the input data per time window slice to effectively improve the performance of time-series forecasting models in the presence of out-of-range data shifts [348]. We employ an adaptive normalization to adjust the operating center signal variations and enable a single AD model for all RBX systems in Refs. [41,70]. The method subtracts the nominal value for each time segment following edge shift detection. We present the renormalization of digi-occupancy to mitigate the impact of collision experiment configuration variations and enhance the generalization of AD modeling for the DQM in Refs. [77,78]. Different temporal modeling tasks may require specific normalization techniques to achieve the desired outcome. Exploring and leveraging an adaptive out-of-range data normalization is necessary to build versatile DL models that are less sensitive to model hyperparameters.

- Large-scale embedding models for knowledge transferring on time series data: The recent surge in pre-trained large language models (LLMs) has grasped the attention of researchers from various industries [323–325]. One of the key benefits is generating robust text embeddings—extracted high-relevance features—that allow model fine-tuning on a specific task using much smaller data sets. Large-scale generic TS data encoding would significantly underpin deep learning efforts across several industries. Such efforts are currently lacking in the time series domain, and most existing models are sensitive to hyperparameters and fragmented [12,81,130]. A recent study called TimeGPT [389] has started exploring generic TS forecasting modeling by training the model on diverse data sets. This is an important development, as experience from LLMs can be applied to TS modeling due to their close relationship in sequence modeling. The TimeGPT study was conducted after completing our Ph.D. and holds promise for the future of TS forecasting. Having a pre-trained TS embedder can greatly accelerate the modeling process in several TS applications.

- Active learning and reinforcement learning for progressing TSAD: Anomaly detection is the process of identifying unusual patterns in data. Unsupervised anomaly detection models detect such outlier patterns without needing labeled data sets. Weakly supervised anomaly detection (WSAD) methods allow for employing incomplete and inexact supervision when obtaining complete and accurate labels for AD tasks is difficult [123]. Two key factors must be noted to build a pragmatic AD model: 1) not all outliers are anomalies, and outliers can be rare regular instances [41], and 2) some anomaly cases might be more interesting than others. AD models should use existing or new knowledge of anomaly patterns to address the above factors. Both unsupervised and weakly supervised approaches rely on one-time trained models and have limited adaptability to new requirements or information after model training. Keeping track of outlier patterns and retraining the AD models can be potential solutions to deal with such variability, but these methods may fall short in large systems with multivariate data. Future research in reinforcement learning and human-in-the-loop active learning would be beneficial for building an end-to-end automating adaptive approach.

- Self-explainable TSAD models using graph-based modeling and attention mechanism: The literature in explainable TSAD is limited, and the existing methods predominantly rely on post-hoc [41, 331] and detection hit matching approaches [15,307]. Future research on graph neural networks [129] and attention mechanisms [185] may potentially enable explanations in quasi-anti-hoc mode. These methods could capture multivariate interaction and temporal saliency while exploiting the power of deep learning.

- Enhancing multivariate anomaly forecasting: We propose AnoP, a multi-horizon anomaly prediction framework, for multivariate time series prognostics in [70]. The efficacy of AnoP relies on the accuracy of its employed TSF and AD models. Adequate anomaly samples are required to train robust anomaly TS forecasting. Anomalies are rare instances constituting much smaller data samples than the normal healthy class samples. The class imbalance can be addressed during the TSF training using weighted cost functions or data augmentation through synthetic data generation to mitigate the learning challenge on limited anomaly samples. The TSF training data can be easily annotated using a previously trained AD model, and higher weights can be assigned to the sections with anomalous patterns during the TSF training loss estimation. The other alternative is to generate and incorporate synthetic data into the training data sets. The recent progress on GAN has demonstrated good capability on multivariate TS signals [131,318]. Further extension of AnoP with the above approaches could be a promising research direction.

Bibliography

[1] Naeem Ahmad Tahir, J Blanco Sancho, Alexander Shutov, Rüdiger Schmidt, and A Roberto Piriz. Impact of high energy high intensity proton beams on targets: case studies for Super Proton Synchrotron and Large Hadron Collider. *Physical Review Special Topics-Accelerators and Beams*, 15(5):051003, 2012.

[2] CERN-CMS detector description. https://home.cern/resources. Accessed: 2010-09-30.

[3] Ettore Focardi. Status of the CMS detector. *Physics Procedia*, 37:119–127, 2012.

[4] David d'Enterria, M Ballintijn, M Bedjidian, D Hofman, O Kodolova, C Loizides, IP Lokthin, C Lourenço, C Mironov, SV Petrushanko, et al. CMS physics technical design report: Addendum on high density QCD with heavy ions. *Journal of Physics G: Nuclear and Particle Physics*, 34(11):2307, 2007.

[5] Izaak Neutelings. CMS coordinate system, 2023. Accessed on 14/12/2023.

[6] J Mans. CMS technical design report for the phase 1 upgrade of the Hadron Calorimeter. Technical report, CMS-TDR-010, 2012.

[7] Pawel de Barbaro, Jay Dittmann, Edward Laird, Danila Tlisov, and The CMS-HCAL Collaboration. HCAL phase 1 lessons. CERN private communications.

[8] Nadja Strobbe. The upgrade of the CMS Hadron Calorimeter with Silicon photomultipliers. *Journal of Instrumentation*, 12(1):C01080, 2017.

[9] Gurpreet Chahal and The CMS-HCAL Collaboration. Data monte carlo preparation in CMS. In *IPPP Organised Workshops and Conferences*. Durham University, 2018.

[10] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: definitions and benchmarks. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[11] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

[12] Nesryne Mejri, Laura Lopez-Fuentes, Kankana Roy, Pavel Chernakov, Enjie Ghorbel, and Djamila Aouada. Unsupervised anomaly detection in time-series: an extensive evaluation and analysis of state-of-the-art methods. *arXiv preprint arXiv:2212.03637*, 2022.

[13] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[15] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. Exathlon: a benchmark for explainable anomaly detection over time series. *arXiv preprint arXiv:2010.05073*, 2020.

[16] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

[17] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[18] Leland McInnes, John Healy, and James Melville. UMAP: uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[19] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963.

[20] Jakob Runge. Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. In *Conference on Uncertainty in Artificial Intelligence*, pages 1388–1397. Proceedings of Machine Learning Research, 2020.

[21] The CMS Collaboration. MUSiC, a model unspecific search for new physics, in pp collisions at $\sqrt{s}$= 8 TeV. *CMS Physics Analysis Summary CMS-PAS-EXO-14/016*, 53, 2017.

[22] Kim Albertsson, Piero Altoe, Dustin Anderson, Michael Andrews, Juan Pedro Araque Espinosa, Adam Aurisano, Laurent Basara, Adrian Bevan, Wahid Bhimji, Daniele Bonacorsi, et al. Machine learning in high energy physics community white paper. In *Journal of Physics: Conference Series*, volume 1085, page 022008. IOP Publishing, 2018.

[23] Javier Duarte and Jean-Roch Vlimant. Graph neural networks for particle tracking and reconstruction. In *Artificial Intelligence for High Energy Physics*, pages 387–436. World Scientific, 2022.

[24] Johan Alwall, Michel Herquet, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. MadGraph 5: going beyond. *Journal of High Energy Physics*, 2011(6):1–40, 2011.

[25] Jack Collins, Kiel Howe, and Benjamin Nachman. Anomaly detection for resonant new physics with machine learning. *Physical Review Letters*, 121(24):241803, 2018.

[26] Oliver Knapp, Olmo Cerri, Günther Dissertori, Thong Q Nguyen, Maurizio Pierini, and Jean-Roch Vlimant. Adversarially learned anomaly detection on CMS open data: re-discovering the top quark. *European Physical Journal Plus*, 136(2):236, 2021.

[27] Jan Hajer, Ying-Ying Li, Tao Liu, and He Wang. Novelty detection meets collider physics. *Physical Review D*, 101(7):076015, 2020.

[28] Thorben Finke, Michael Krämer, Alessandro Morandini, Alexander Mück, and Ivan Oleksiyuk. Autoencoders for unsupervised anomaly detection in high energy physics. *Journal of High Energy Physics*, 2021(6):1–32, 2021.

[29] Oliver Atkinson, Akanksha Bhardwaj, Christoph Englert, Vishal S Ngairangbam, and Michael Spannowsky. Anomaly detection with convolutional graph neural networks. *Journal of High Energy Physics*, 2021(8):1–19, 2021.

[30] Layne Bradshaw, Spencer Chang, and Bryan Ostdiek. Creating simple, interpretable anomaly detectors for new physics in jet substructure. *Physical Review D*, 106(3):035014, 2022.

[31] Katherine Fraser, Samuel Homiller, Rashmish K Mishra, Bryan Ostdiek, and Matthew D Schwartz. Challenges for unsupervised anomaly detection in particle physics. *Journal of High Energy Physics*, 2022(3):1–31, 2022.

[32] Taoli Cheng, Jean-François Arguin, Julien Leissner-Martin, Jacinthe Pilette, and Tobias Golling. Variational autoencoders for anomalous jet tagging. *Physical Review D*, 107(1):016002, 2023.

[33] Thorsten Buss, Barry M Dillon, Thorben Finke, Michael Krämer, Alessandro Morandini, Alexander Mück, Ivan Oleksiyuk, and Tilman Plehn. What's anomalous in LHC jets? *SciPost Physics*, 49:50, 2022.

[34] Barry M Dillon, Radha Mastandrea, and Benjamin Nachman. Self-supervised anomaly detection for new physics. *Physical Review D*, 106(5):056005, 2022.

[35] Vishal S Ngairangbam, Michael Spannowsky, and Michihisa Takeuchi. Anomaly detection in high-energy physics using a quantum autoencoder. *Physical Review D*, 105(9):095004, 2022.

[36] Sulaiman Alvi, Christian W Bauer, and Benjamin Nachman. Quantum anomaly detection for collider physics. *Journal of High Energy Physics*, 2023(2):1–17, 2023.

[37] Matthew Feickert and Benjamin Nachman. A living review of machine learning for particle physics. *arXiv preprint arXiv:2102.02770*, 2023.

[38] J Arjona Martínez, Olmo Cerri, Maria Spiropulu, JR Vlimant, and M Pierini. Pileup mitigation at the Large Hadron Collider with graph neural networks. *European Physical Journal Plus*, 134(7):333, 2019.

[39] Virginia Azzolin, Michael Andrews, Gianluca Cerminara, Nabarun Dev, Colin Jessop, Nancy Marinelli, Tanmay Mudholkar, Maurizio Pierini, Adrian Pol, and Jean-Roch Vlimant. Improving data quality monitoring via a partnership of technologies and resources between the CMS experiment at CERN and industry. In *European Physical Journal Web of Conferences*, volume 214, page 01007. EDP Sciences, 2019.

[40] Adrian Alan Pol, Gianluca Cerminara, Cécile Germain, Maurizio Pierini, and Agrima Seth. Detector monitoring with artificial neural networks at the CMS experiment at the CERN Large Hadron Collider. *Computing and Software for Big Science*, 3(1):3, 2019.

[41] Mulugeta Weldezgina Asres, Grace Cummings, Pavel Parygin, Aleko Khukhunaishvili, Maria Toms, Alan Campbell, Seth I Cooper, David Yu, Jay Dittmann, and Christian W Omlin. Unsupervised deep variational model for multivariate sensor anomaly detection. In *International Conference on Progress in Informatics and Computing*, pages 364–371. IEEE, 2021.

[42] Sea Agostinelli, John Allison, K al Amako, John Apostolakis, H Araujo, Pedro Arce, Makoto Asai, D Axen, Swagato Banerjee, GJNI Barrand, et al. GEANT4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[43] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. A brief introduction to PYTHIA 8.1. *Computer Physics Communications*, 178(11):852–867, 2008.

[44] Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to LHC physics. *Annual Review of Nuclear and Particle Science*, 68:161–181, 2018.

[45] Raghav Kansal, Javier Duarte, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. Graph generative adversarial networks for sparse data generation in high energy physics. *arXiv preprint arXiv:2012.00173*, 2020.

[46] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Accelerating science with generative adversarial networks: an application to 3D particle showers in multilayer calorimeters. *Physical Review Letters*, 120(4):042003, 2018.

[47] Chinmaya Mahesh, Kristin Dona, David W Miller, and Yuxin Chen. Towards an interpretable data-driven trigger system for high-throughput physics facilities. *arXiv preprint arXiv:2104.06622*, 2021.

[48] Joosep Pata, Javier Duarte, Farouk Mokhtar, Eric Wulff, Jieun Yoo, Jean-Roch Vlimant, Maurizio Pierini, Maria Girone, and CMS Collaboration. Machine learning for particle flow reconstruction at CMS. In *Journal of Physics: Conference Series*, volume 2438, page 012100. IOP Publishing, 2023.

[49] Virginia Azzolini, Dmitrijus Bugelskis, Tomas Hreus, Kaori Maeshima, Menendez Javier Fernandez, Antanas Norkus, Patrick James Fraser, Marco Rovere, Marcel Andre Schneider, et al. The data quality monitoring software for the CMS experiment at the LHC: past, present and future. In *European Physical Journal Web of Conferences*, volume 214, page 02003, 2019.

[50] Adrian Alan Pol, Virginia Azzolini, Gianluca Cerminara, Federico De Guio, Giovanni Franzoni, Maurizio Pierini, Filip Širokỳ, and Jean-Roch Vlimant. Anomaly detection using deep autoencoders for the assessment of the quality of the data acquired by the CMS experiment. In *European Physical Journal Web of Conferences*, volume 214, page 06008. EDP Sciences, 2019.

[51] Adrian Alan Pol, Victor Berger, Cecile Germain, Gianluca Cerminara, and Maurizio Pierini. Anomaly detection with conditional variational autoencoders. In *International Conference on Machine Learning and Applications*, pages 1651–1657. IEEE, 2019.

[52] Maciej Wielgosz, Matej Mertik, Andrzej Skoczeń, and Ernesto De Matteis. The model of an anomaly detector for HiLumi LHC magnets based on recurrent neural networks and adaptive quantization. *Engineering Applications of Artificial Intelligence*, 74:166–185, 2018.

[53] Armin Halilovic. *Anomaly detection for the CERN Large Hadron Collider injection magnets*. PhD thesis, KU Leuven, 2018.

[54] Thiebout Thomas Dewitte. Anomaly detection for the injection kickermagnets of the CERN Large Hadron Collider. CERN-THESIS-2019-294, 2019.

[55] Maciej Wielgosz, Andrzej Skoczeń, and Matej Mertik. Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 867:40–50, 2017.

[56] Maciej Wielgosz, Andrzej Skoczen, and Kazimierz Wiatr. Looking for a correct solution of anomaly detection in the LHC machine protection system. In *International Conference on Signals and Electronic Systems*, pages 257–262. IEEE, 2018.

[57] Lyndon Evans and Philip Bryant. LHC machine. *Journal of instrumentation*, 3(08):S08001, 2008.

[58] Rolf-Dieter Heuer. The future of the Large Hadron Collider and CERN. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1961):986–994, 2012.

[59] The CMS Collaboration, S Chatrchyan, G Hmayakyan, V Khachatryan, AM Sirunyan, W Adam, T Bauer, T Bergauer, H Bergauer, M Dragicevic, et al. The CMS experiment at the CERN LHC. *Journal of Instrumentation*, 3:S08004, 2008.

[60] Serguei Chatrchyan, Vardan Khachatryan, Albert M Sirunyan, Armen Tumasyan, Wolfgang Adam, Ernest Aguilo, Thomas Bergauer, M Dragicevic, J Erö, C Fabjan, et al. Observation of a new boson at a mass of 125 gev with the CMS experiment at the LHC. *Physics Letters B*, 716(1):30–61, 2012.

[61] The CMS Collaboration. Precise determination of the mass of the Higgs boson and tests of compatibility of its couplings with the standard model predictions using proton collisions at 7 and 8 tev. *European Physical Journal C*, 75(5):212, 2015.

[62] Wolfgang Adam, T Bergauer, C Deldicque, J Ero, R Fruehwirth, M Jeitler, K Kastner, S Kostner, N Neumeister, M Padrta, et al. The CMS high level trigger. *European Physical Journal C*, 46(3), 2006.

[63] S Abdullin, V Abramov, B Acharya, M Adams, N Akchurin, U Akgun, EW Anderson, G Antchev, S Ayan, S Aydin, et al. Design, performance, and calibration of CMS Hadron-Barrel calorimeter wedges. *The European Physical Journal C*, 55:159–171, 2008.

[64] G Baiatian, Majid Hashemi, Sharon Hagopian, Andras Pal, Emanuel Machado, Engin Isiksal, V ODell, T Haelen, David A Sanders, Efe Yazgan, et al. Design, performance, and calibration of CMS Hadron Endcap calorimeters. Technical report, CERN-CMS-NOTE-2008-010, 2008.

[65] Salavat Abdullin, Victor Abramov, B Acharya, Nadia Adam, M Adams, Nural Akchurin, Ugur Akgun, E Albayrak, E Walter Anderson, Georgy Antchev, et al. Design, performance, and calibration of the CMS Hadron-Outer calorimeter. *The European Physical Journal C*, 57:653–663, 2008.

[66] S Abdullin, V Abramov, B Acharya, M Adams, N Akchurin, U Akgun, EW Anderson, G Antchev, M Arcidy, S Ayan, et al. Design, performance, and calibration of CMS forward calorimeter wedges. *The European Physical Journal C*, 53:139–166, 2008.

[67] M Schneider. The data quality monitoring software for the CMS experiment at the LHC: past, present and future. In *International Conference on Computing in High Energy and Nuclear Physics*, 2018.

[68] Grace Cummings and The CMS Collaboration. CMS HCAL VTRx-induced communication loss and mitigation. *Journal of Instrumentation*, 17(05):C05020, 2022.

[69] Matteo Paltenghi. *Time series anomaly detection for CERN large-scale computing infrastructure*. PhD thesis, Polytechnic of Milan, 2020.

[70] Mulugeta Weldezgina Asres, Grace Cummings, Aleko Khukhunaishvili, Pavel Parygin, Seth I Cooper, David Yu, Jay Dittmann, and Christian W Omlin. Long horizon anomaly prediction in multivariate time series with causal autoencoders. In *PHM Society European Conference*, volume 7, pages 21–31, 2022.

[71] Lassi Tuura, A Meyer, Ilaria Segoni, and G Della Ricca. CMS data quality monitoring: systems and experiences. In *Journal of Physics: Conference Series*, volume 219, page 072020. IOP Publishing, 2010.

[72] Federico De Guio and The CMS Collaboration. The CMS data quality monitoring software: experience and future prospects. In *Journal of Physics: Conference Series*, volume 513, page 032024. IOP Publishing, 2014.

[73] Mulugeta Weldezgina Asres, Christian W. Omlin, Jay Dittmann, Pavel Parygin, and The CMS-HCAL Collaboration. Lightweight mechanism for multi-system interconnection and divergence discovery. *The CMS Detector Note: DN2023/031*, 2023.

[74] Mulugeta Weldezgina Asres, Christian W. Omlin, Jay Dittmann, Pavel Parygin, and The CMS-HCAL Collaboration. Scalable online anomaly causality discovery for large complex systems at the cms experiment. *The CMS Detector Note: DN2023/030*, 2023.

[75] Oleksandr Viazlo and The CMS Collaboration. Non-uniformity in HE digi-occupancy distributions. CERN private communications, 2022.

[76] Mulugeta Weldezgina Asres, Christian W. Omlin, Long Wang, and David Yu. Spatio-temporal anomaly detection for the DQM of the CMS experiment via graph networks. In *Inter-experiment Machine Learning Workshop*. CERN, 2022.

[77] Mulugeta Weldezgina Asres, Christian Walter Omlin, Long Wang, David Yu, Pavel Parygin, Jay Dittmann, Georgia Karapostoli, Markus Seidel, Rosamaria

Venditti, Luka Lambrecht, et al. Spatio-temporal anomaly detection with graph networks for data quality monitoring of the Hadron Calorimeter. *Sensors*, 23(24):9679, 2023.

[78] Mulugeta Weldezgina Asres, Christian W. Omlin, Long Wang, David Yu, Jay Dittmann, and The CMS-HCAL Collaboration. Extending anomaly detection for the data quality monitoring through transfer learning between calorimeters. *The CMS Detector Note: DN2023/023*, 2023.

[79] Senzhang Wang, Hao Miao, Jiyue Li, and Jiannong Cao. Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(5):4695–4705, 2021.

[80] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: a survey. *arXiv:1901.03407*, 2019.

[81] Yan Zhao, Liwei Deng, Xuanhao Chen, Chenjuan Guo, Bin Yang, Tung Kieu, Feiteng Huang, Torben Bach Pedersen, Kai Zheng, and Christian S Jensen. A comparative study on unsupervised anomaly detection for time series: experiments and analysis. *arXiv preprint arXiv:2209.04635*, 2022.

[82] Andrew A Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for IoT time-series data: a survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2019.

[83] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[84] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[85] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. Proceedings of Machine Learning Research, 2017.

[86] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996, 2019.

[87] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: a review. *ACM Computing Surveys*, 54(2):1–38, 2021.

[88] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. *IEEE Access*, 2021.

[89] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104. ACM, 2000.

[90] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[91] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner. Extended isolation forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489, 2019.

[92] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop*, pages 172–179. IEEE Press, 2003.

[93] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in Neural Information Processing Systems*, 12, 1999.

[94] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 427–438. ACM, 2000.

[95] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. *arXiv preprint arXiv:2009.02040*, 2020.

[96] Jake Cowton, Ilias Kyriazakis, Thomas Plötz, and Jaume Bacardit. A combined deep learning GRU-autoencoder for the early detection of respiratory disease in pigs using multiple environmental sensors. *Sensors*, 18(8):2521, 2018.

[97] Yifan Guo, Weixian Liao, Qianlong Wang, Lixing Yu, Tianxi Ji, and Pan Li. Multidimensional time series anomaly detection: A GRU-based Gaussian mixture variational autoencoder approach. In *Asian Conference on Machine Learning*, pages 97–112. Proceedings of Machine Learning Research, 2018.

[98] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 1409–1416. Association for the Advancement of Artificial Intelligence, 2019.

[99] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7:1991–2005, 2018.

[100] Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. Multi-head CNN–RNN for multi-time series anomaly detection: an industrial case study. *Neurocomputing*, 363:246–260, 2019.

[101] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M Shamim Hossain. Deep anomaly detection for time-series data in industrial IoT: a communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 2020.

[102] Jonguk Kim, Jeong-Han Yun, and Hyoung Chun Kim. Anomaly detection for industrial control systems using sequence-to-sequence neural networks. In *Computer Security*, pages 3–18. Springer, 2019.

[103] Zijian Niu, Ke Yu, and Xiaofei Wu. LSTM-based vae-gan for time-series anomaly detection. *Sensors*, 20(13):3738, 2020.

[104] Yeji Choi, Hyunki Lim, Heeseung Choi, and Ig-Jae Kim. GAN-based anomaly detection and localization of multivariate time series data for power plant. In *BigComp*, pages 71–74. IEEE, 2020.

[105] Dan Li, Dacheng Chen, Jonathan Goh, and See-kiong Ng. Anomaly detection with generative adversarial networks for multivariate time series. *arXiv preprint arXiv:1809.04758*, 2018.

[106] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. BeatGAN: anomalous rhythm detection using adversarially generated time series. In *International Joint Conference on Artificial Intelligence*, pages 4433–4439, 2019.

[107] Liwei Deng, Xuanhao Chen, Yan Zhao, and Kai Zheng. HIFI: anomaly detection for multivariate time series with high-order feature interactions. In *International Conference on Database Systems for Advanced Applications*, pages 641–649. Springer, 2021.

[108] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 4027–4035. Association for the Advancement of Artificial Intelligence, 2021.

[109] Gangqiang Zhang, Wei Zheng, Wenjie Yin, and Weiwei Lei. Improving the resolution and accuracy of groundwater level anomalies using the machine learning-based fusion model in the north China plain. *Sensors*, 21(1):46, 2020.

[110] David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. Graph-based deep learning for medical diagnosis and analysis: past, present and future. *Sensors*, 21(14):4758, 2021.

[111] Daniel Hsu. Anomaly detection on graph time series. *arXiv preprint arXiv:1708.02975*, 2017.

[112] Leyan Deng, Defu Lian, Zhenya Huang, and Enhong Chen. Graph convolutional adversarial networks for spatiotemporal anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2416–2428, 2022.

[113] Zorana Banković, David Fraga, José M Moya, and Juan Carlos Vallejo. Detecting unknown attacks in wireless sensor networks that contain mobile nodes. *Sensors*, 12(8):10834–10850, 2012.

[114] Leo Tišljarić, Sofia Fernandes, Tonči Carić, and João Gama. Spatiotemporal road traffic anomaly detection: a tensor-based approach. *Applied Sciences*, 11(24):12017, 2021.

[115] Lin Jiang, Hang Xu, Jinhai Liu, Xiangkai Shen, Senxiang Lu, and Zhan Shi. Anomaly detection of industrial multi-sensor signals based on enhanced spatiotemporal features. *Neural Computing and Applications*, pages 1–13, 2022.

[116] Dan Xu, Yan Yan, Elisa Ricci, and Nicu Sebe. Detecting anomalous events in videos by learning deep representations of appearance and motion. *Computer Vision and Image Understanding*, 156:117–127, 2017.

[117] Yunpeng Chang, Zhigang Tu, Wei Xie, Bin Luo, Shifu Zhang, Haigang Sui, and Junsong Yuan. Video anomaly detection with spatio-temporal dissociation. *Pattern Recognition*, 122:108213, 2022.

[118] Weixin Luo, Wen Liu, Dongze Lian, Jinhui Tang, Lixin Duan, Xi Peng, and Shenghua Gao. Video anomaly detection with sparse coding inspired deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(3):1070–1084, 2019.

[119] Peng Wu, Jing Liu, Mingming Li, Yujia Sun, and Fang Shen. Fast sparse coding networks for anomaly detection in videos. *Pattern Recognition*, 107:107515, 2020.

[120] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. In *Proceedings of Computer Vision and Pattern Recognition*, pages 733–742. IEEE, 2016.

[121] Waseem Ullah, Amin Ullah, Tanveer Hussain, Zulfiqar Ahmad Khan, and Sung Wook Baik. An efficient anomaly recognition framework using an attention residual LSTM in surveillance videos. *Sensors*, 21(8):2811, 2021.

[122] Anton Yeshchenko, Claudio Di Ciccio, Jan Mendling, and Artem Polyvyanyy. Comprehensive process drift detection with visual analytics. In *International Conference on Conceptual Modeling*, pages 119–135. Springer, 2019.

[123] Minqi Jiang, Chaochuan Hou, Ao Zheng, Xiyang Hu, Songqiao Han, Hailiang Huang, Xiangnan He, Philip S Yu, and Yue Zhao. Weakly supervised anomaly detection: a survey. *arXiv preprint arXiv:2302.04549*, 2023.

[124] Yue Zhao, Xiyang Hu, Cheng Cheng, Cong Wang, Changlin Wan, Wen Wang, Jianing Yang, Haoping Bai, Zheng Li, Cao Xiao, et al. Suod: accelerating large-scale unsupervised heterogeneous outlier detection. In *Proceedings of Machine Learning and Systems*, volume 3, pages 463–478, 2021.

[125] Suwon Suh, Daniel H Chae, Hyon-Goo Kang, and Seungjin Choi. Echo-state conditional variational autoencoder for anomaly detection. In *International Joint Conference on Neural Networks*, pages 1015–1022. IEEE, 2016.

[126] Tailai Wen and Roy Keyes. Time series anomaly detection using convolutional neural networks and transfer learning. *arXiv preprint arXiv:1905.13628*, 2019.

[127] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716. Springer, 2019.

[128] Q He, YJ Zheng, CL Zhang, and HY Wang. MTAD-TF: multivariate time series anomaly detection using the combination of temporal pattern and feature pattern. *Complexity*, 2020, 2020.

[129] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. Learning graph structures with transformer for multivariate time series anomaly detection in IoT. *IEEE Internet of Things Journal*, 2021.

[130] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022.

[131] Mélanie Ducoffe, Ilyass Haloui, and Jayant Sen Gupta. Anomaly detection on time series with wasserstein GAN applied to PHM. *International Journal of Prognostics and Health Management*, 10(4), 2019.

[132] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. TadGAN: time series anomaly detection using generative adversarial networks. *arXiv preprint arXiv:2009.07769*, 2020.

[133] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference in Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[134] Dong-Hoon Shin, Roy C Park, and Kyungyong Chung. Decision boundary-based anomaly detection model using improved AnoGAN from ECG data. *IEEE Access*, 8:108664–108674, 2020.

[135] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021.

[136] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. TranAD: deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022.

[137] Xixuan Wang, Dechang Pi, Xiangyan Zhang, Hao Liu, and Chang Guo. Variational transformer-based anomaly detection approach for multivariate time series. *Measurement*, 191:110791, 2022.

[138] Hongwei Zhang, Yuanqing Xia, Tijin Yan, and Guiyang Liu. Unsupervised anomaly detection in multivariate time series through transformer-based variational autoencoder. In *Chinese Control and Decision Conference*, pages 281–286. IEEE, 2021.

[139] Desen Huang, Lifeng Shen, Zhongzhong Yu, Zhenjing Zheng, Min Huang, and Qianli Ma. Efficient time series anomaly detection by multiresolution self-supervised discriminative network. *Neurocomputing*, 491:261–272, 2022.

[140] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. *Advances in Neural Information Processing Systems*, 31, 2018.

[141] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. Spatio-temporal data mining: a survey of problems and methods. *ACM Computing Surveys*, 51(4):1–41, 2018.

[142] Jingtao Hu, En Zhu, Siqi Wang, Xinwang Liu, Xifeng Guo, and Jianping Yin. An efficient and robust unsupervised anomaly detection method using ensemble random projection in surveillance videos. *Sensors*, 19(19):4145, 2019.

[143] Xiang Li, Wei Zhang, Qian Ding, and Jian-Qiao Sun. Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation. *Journal of Intelligent Manufacturing*, 31(2):433–452, 2020.

[144] Kai Zhou and Jiong Tang. Harnessing fuzzy neural network for gear fault diagnosis with limited data labels. *The International Journal of Advanced Manufacturing Technology*, 115(4):1005–1019, 2021.

[145] Tiancheng Shi, Yigang He, Tao Wang, and Bing Li. Open switch fault diagnosis method for PWM voltage source rectifier based on deep learning approach. *IEEE Access*, 7:66595–66608, 2019.

[146] Mihalj Bakator and Dragica Radosav. Deep learning and medical diagnosis: a review of literature. *Multimodal Technologies and Interaction*, 2(3):47, 2018.

[147] Madalina-Mihaela Buzau, Javier Tejedor-Aguilera, Pedro Cruz-Romero, and Antonio Gómez-Expósito. Hybrid deep neural networks for detection of non-technical losses in electricity smart meters. *IEEE Transactions on Power Systems*, 35(2):1254–1263, 2019.

[148] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at Microsoft. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3009–3017. ACM, 2019.

[149] Kamran Shaukat, Talha Mahboob Alam, Suhuai Luo, Shakir Shabbir, Ibrahim A Hameed, Jiaming Li, Syed Konain Abbas, and Umair Javed. A review of time-series anomaly detection techniques: a step to future perspectives. In *Advances in Information and Communication: Proceedings of the Future of Information and Communication Conference*, volume 1, pages 865–877. Springer, 2021.

[150] Xiaodi Hou and Liqing Zhang. Saliency detection: a spectral residual approach. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[151] Wei Lu and Ali A Ghorbani. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing*, 2009:1–16, 2008.

[152] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1067–1075. ACM, 2017.

[153] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

[154] Pooja Kamat and Rekha Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S Web of Conferences*, volume 170, page 02007. EDP Sciences, 2020.

[155] Carolin Wagner and Bernd Hellingrath. Supporting the implementation of predictive maintenance: a process reference model. *International Journal of Prognostics and Health Management*, 12(1), 2021.

[156] Rocco Langone, Alfredo Cuzzocrea, and Nikolaos Skantzos. Interpretable anomaly prediction: predicting anomalous behavior in industry 4.0 settings via regularized logistic regression tools. *Data and Knowledge Engineering*, 130:101850, 2020.

[157] Chao Huang, Xian Wu, and Dong Wang. Crowdsourcing-based urban anomaly prediction system for smart cities. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 1969–1972. ACM, 2016.

[158] Jianwu Wang, Chen Liu, Meiling Zhu, Pei Guo, and Yapeng Hu. Sensor data based system-level anomaly prediction for smart manufacturing. In *BigData*, pages 158–165. IEEE, 2018.

[159] Imed Hadj-Kacem, Sana Ben Jemaa, Sylvain Allio, and Yosra Ben Slimen. Anomaly prediction in mobile networks: a data driven approach for machine learning algorithm selection. In *Network Operations and Management Symposium*, pages 1–7. IEEE, 2020.

[160] Jingwen Wang, Jingxin Liu, Juntao Pu, Qinghong Yang, Zhongchen Miao, Jian Gao, and You Song. An anomaly prediction framework for financial it systems using hybrid machine learning methods. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10, 2019.

[161] Seyed Mohammad Rezvanizaniani, Jacob Dempsey, and Jay Lee. An effective predictive maintenance approach based on historical maintenance data using a probabilistic risk assessment: PHM14 data challenge. *International Journal of Prognostics and Health Management*, 5(2), 2014.

[162] Zhiyi Tang, Zhicheng Chen, Yuequan Bao, and Hui Li. Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring. *Structural Control and Health Monitoring*, 26(1):e2296, 2019.

[163] Valentin Hamaide and François Glineur. Unsupervised minimum redundancy maximum relevance feature selection for predictive maintenance: application to a rotating machine. *International Journal of Prognostics and Health Management*, 12(2), 2021.

[164] Narendhar Gugulothu, Vishnu Tv, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Predicting remaining useful life using time series embeddings based on recurrent neural networks. *arXiv:1709.01073*, 2017.

[165] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: a systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[166] Hao-Fan Yang and Yi-Ping Phoebe Chen. Hybrid deep learning and empirical mode decomposition model for time series applications. *Expert Systems with Applications*, 120:128–138, 2019.

[167] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2021.

[168] Yeqi Liu, Chuanyang Gong, Ling Yang, and Yingyi Chen. DSTP-RNN: a dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. *Expert Systems with Applications*, 143:113082, 2020.

[169] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv:1711.11053*, 2017.

[170] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv:1707.01926*, 2017.

[171] Yagmur Gizem Cinar, Hamid Mirisaee, Parantapa Goswami, Eric Gaussier, Ali Aït-Bachir, and Vadim Strijov. Position-based content attention for time series forecasting with sequence-to-sequence RNNs. In *Neural Information Processing: International Conference*, pages 533–544. Springer, 2017.

[172] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv:1704.02971*, 2017.

[173] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[174] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[175] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in Neural Information Processing Systems*, 33:17766–17778, 2020.

[176] Zongying Liu, Chu Kiong Loo, and Kitsuchart Pasupa. A novel error-output recurrent two-layer extreme learning machine for multi-step time series prediction. *Sustainable Cities and Society*, 66:102613, 2021.

[177] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: multivariate time series forecasting with graph neural networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 753–763. ACM, 2020.

[178] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[179] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[180] Yang Lin, Irena Koprinska, and Mashud Rana. SSDNet: State space decomposition neural network for time series forecasting. In *International Conference on Data Mining*, pages 370–378. IEEE, 2021.

[181] Binh Tang and David S Matteson. Probabilistic transformer for time series analysis. *Advances in Neural Information Processing Systems*, 34:23592–23608, 2021.

[182] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2021.

[183] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. *Advances in Neural Information Processing Systems*, 33:17105–17115, 2020.

[184] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: frequency enhanced decomposed transformer for long-term series forecasting. *arXiv preprint arXiv:2201.12740*, 2022.

[185] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: a survey. *arXiv preprint arXiv:2202.07125*, 2022.

[186] Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. A comparison of transformer and LSTM encoder-decoder models for ASR. In *Automatic Speech Recognition and Understanding Workshop*, pages 8–15. IEEE, 2019.

[187] Siyu Shao, Stephen McAleer, Ruqiang Yan, and Pierre Baldi. Highly accurate machine fault diagnosis using deep transfer learning. *IEEE Transactions on Industrial Informatics*, 15(4):2446–2455, 2018.

[188] Nikolay Laptev, Jiafan Yu, and Ram Rajagopal. Reconstruction and regression loss for time-series transfer learning. In *Proceedings of the Special Interest Group on SIGKDD Knowledge Discovery and Data Mining*, volume 20, 2018.

[189] Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Transfer learning for clinical time series analysis using recurrent neural networks. *arXiv preprint arXiv:1807.01705*, 2018.

[190] Nicolas Boullé, Vassilios Dallas, Yuji Nakatsukasa, and D Samaddar. Classification of chaotic time series with deep learning. *Physica D: Nonlinear Phenomena*, 403:132261, 2020.

[191] Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1019–1034, 2014.

[192] Peng Xiong, Yonxin Zhu, Zhanrui Sun, Zihao Cao, Menglin Wang, Yu Zheng, Junjie Hou, Tian Huang, and Zhiqiang Que. Application of transfer learning in continuous time series for anomaly detection in commercial aircraft flight data. In *International Conference on Smart Cloud*, pages 13–18. IEEE, 2018.

[193] Shuteng Niu, Yongxin Liu, Jian Wang, and Houbing Song. A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, 1(2):151–166, 2020.

[194] Leye Wang, Xu Geng, Xiaojuan Ma, Feng Liu, and Qiang Yang. Cross-city transfer learning for deep spatio-temporal prediction. *arXiv preprint arXiv:1802.00386*, 2018.

[195] Bin Guo, Jing Li, Vincent W Zheng, Zhu Wang, and Zhiwen Yu. CityTransfer: transferring inter-and intra-city knowledge for chain store site recommendation based on multi-source urban data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–23, 2018.

[196] Mei Wang and Weihong Deng. Deep visual domain adaptation: a survey. *Neurocomputing*, 312:135–153, 2018.

[197] Ruoying Wang, Kexin Nie, Tie Wang, Yang Yang, and Bo Long. Deep learning for anomaly detection. In *Proceedings of International Conference on Web Search and Data Mining*, pages 894–896, 2020.

[198] Peng Xie, Tianrui Li, Jia Liu, Shengdong Du, Xin Yang, and Junbo Zhang. Urban flow prediction from spatiotemporal data using machine learning: a survey. *Information Fusion*, 59:1–12, 2020.

[199] Ruocheng Guo, Lu Cheng, Jundong Li, P Richard Hahn, and Huan Liu. A survey of learning causality with data: problems and methods. *ACM Computing Surveys*, 53(4):1–37, 2020.

[200] Chao Liu, Kin Gwn Lore, Zhanhong Jiang, and Soumik Sarkar. Root-cause analysis for time-series anomalies via spatiotemporal graphical modeling in distributed complex systems. *Knowledge-Based Systems*, 211:106527, 2021.

[201] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. CloudRanger: root cause identification for cloud native systems. In *International Symposium on Cluster, Cloud and Grid Computing*, pages 492–502. IEEE, 2018.

[202] Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. Localizing failure root causes in a microservice through causality inference. In *International Symposium on Quality of Service*, pages 1–10. IEEE, 2020.

[203] Bram Steenwinckel, Dieter De Paepe, Sander Vanden Hautte, Pieter Heyvaert, Mohamed Bentefrit, Pieter Moens, Anastasia Dimou, Bruno Van Den Bossche, Filip De Turck, Sofie Van Hoecke, et al. FLAGS: a methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. *Future Generation Computer Systems*, 116:30–48, 2021.

[204] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[205] Mathieu Chevalley, Yusuf Roohani, Arash Mehrjou, Jure Leskovec, and Patrick Schwab. CausalBench: a large-scale benchmark for network inference from single-cell perturbation data. *arXiv preprint arXiv:2210.17283*, 2022.

[206] Chainarong Amornbunchornvej, Navaporn Surasvadi, Anon Plangprasopchok, and Suttipong Thajchayapong. Framework for inferring empirical causal graphs from binary data to support multidimensional poverty analysis. *Heliyon*, 9(5), 2023.

[207] Kailash Budhathoki, Lenon Minorics, Patrick Blöbaum, and Dominik Janzing. Causal structure-based root cause analysis of outliers. In *International Conference on Machine Learning*, pages 2357–2369. Proceedings of Machine Learning Research, 2022.

[208] Charles K Assaad, Imad Ez-Zejjari, and Lei Zan. Root cause identification for collective anomalies in time series given an acyclic summary causal graph with loops. In *International Conference on Artificial Intelligence and Statistics*, pages 8395–8404. Proceedings of Machine Learning Research, 2023.

[209] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10:524, 2019.

[210] Jiawei Chen and Chunhui Zhao. Multi-lag and multi-type temporal causality inference and analysis for industrial process fault diagnosis. *Control Engineering Practice*, 124:105174, 2022.

[211] Chang Tian, Chunhui Zhao, Haidong Fan, and Zhenwei Zhang. Causal network construction based on convergent cross mapping (ccm) for alarm system root cause tracing of nonlinear industrial process. *IFAC-PapersOnLine*, 53(2):13619–13624, 2020.

[212] Qiming Chen, Xun Lang, Shan Lu, Naveed ur Rehman, Lei Xie, and Hongye Su. Detection and root cause analysis of multiple plant-wide oscillations using multivariate nonlinear chirp mode decomposition and multivariate Granger causality. *Computers and Chemical Engineering*, 147:107231, 2021.

[213] Bahador Rashidi, Dheeraj Sharan Singh, and Qing Zhao. Data-driven root-cause fault diagnosis for multivariate non-linear processes. *Control Engineering Practice*, 70:134–147, 2018.

[214] Kai Qin, Lei Chen, Jintao Shi, Zhenxing Li, and Kuangrong Hao. Root cause analysis of industrial faults based on binary extreme gradient boosting and temporal causal discovery network. *Chemometrics and Intelligent Laboratory Systems*, 225:104559, 2022.

[215] Nan Liu, Minggang Hu, Ji Wang, Yujia Ren, and Wende Tian. Fault detection and diagnosis using Bayesian network model combining mechanism correlation analysis and process data: application to unmonitored root cause variables type faults. *Process Safety and Environmental Protection*, 164:15–29, 2022.

[216] Dieter De Paepe, Sander Vanden Hautte, Bram Steenwinckel, Filip De Turck, Femke Ongenae, Olivier Janssens, and Sofie Van Hoecke. A generalized matrix profile framework with support for contextual series analysis. *Engineering Applications of Artificial Intelligence*, 90:103487, 2020.

[217] Viktor Leonhardt, Felix Claus, and Christoph Garth. PEN: process estimator neural network for root cause analysis using graph convolution. *Journal of Manufacturing Systems*, 62:886–902, 2022.

[218] Yujie Zhou, Ke Xu, and Fei He. Root cause diagnosis in multivariate time series based on modified temporal convolution and multi-head self-attention. *Journal of Process Control*, 117:14–25, 2022.

[219] Meike Nauta, Doina Bucur, and Christin Seifert. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):312–340, 2019.

[220] Kailash Budhathoki, Dominik Janzing, Patrick Bloebaum, and Hoiyi Ng. Why did the distribution change? In *International Conference on Artificial Intelligence and Statistics*, pages 1666–1674. Proceedings of Machine Learning Research, 2021.

[221] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc., 2017.

[222] Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michele Sebag. Learning functional causal models with generative neural networks. *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 39–80, 2018.

[223] Dimitris Margaritis et al. *Learning Bayesian network model structure from data.* PhD thesis, School of Computer Science, Carnegie Mellon University Pittsburgh, PA, USA, 2003.

[224] Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale Markov blanket discovery. In *Florida AI Research Society Conference*, volume 2, pages 376–380. American Association for Artificial Intelligence, 2003.

[225] Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 673–678. ACM, 2003.

[226] Diego Colombo, Marloes H Maathuis, et al. Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

[227] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006.

[228] Andreas Gerhardus and Jakob Runge. High-recall causal discovery for autocorrelated time series with latent confounders. *Advances in Neural Information Processing Systems*, 33:12615–12625, 2020.

[229] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[230] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[231] Joseph D Ramsey. Scaling up greedy causal search for continuous variables. *arXiv preprint arXiv:1507.07749*, 2015.

[232] Juan Miguel Ogarrio, Peter Spirtes, and Joe Ramsey. A hybrid causal search algorithm for latent variable models. In *Conference on Probabilistic Graphical Models*, pages 368–379. Proceedings of Machine Learning Research, 2016.

[233] Peter Bühlmann, Jonas Peters, and Jan Ernest. CAM: causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42:2526–2556, 2014.

[234] Yi Liu, Han-Sheng Chen, Haibin Wu, Yun Dai, Yuan Yao, and Zhengbing Yan. Simplified Granger causality map for data-driven root cause diagnosis of process disturbances. *Journal of Process Control*, 95:45–54, 2020.

[235] Ali Shojaie and Emily B Fox. Granger causality: a review and recent advances. *Annual Review of Statistics and Its Application*, 9:289–319, 2022.

[236] Jakob Runge. Causal network reconstruction from time series: from theoretical assumptions to practical estimation. *Chaos: an Interdisciplinary Journal of Nonlinear Science*, 28(7), 2018.

[237] Elena Saggioro, Jana de Wiljes, Marlene Kretschmer, and Jakob Runge. Reconstructing regime-dependent causal relationships from observational time series. *Chaos: an Interdisciplinary Journal of Nonlinear Science*, 30(11), 2020.

[238] Rahul Biswas and Eli Shlizerman. Statistical perspective on functional and causal neural connectomics: the time-aware pc algorithm. *PLOS Computational Biology*, 18(11):e1010653, 2022.

[239] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[240] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.

[241] Richard E Neapolitan et al. *Learning Bayesian networks*, volume 38. Pearson Prentice Hall, 2004.

[242] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: a deep neural network approach to high-dimensional time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[243] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[244] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[245] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[246] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[247] Diganta Misra. Mish: a self-regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[248] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[249] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, volume 37, pages 448–456. Proceedings of Machine Learning Research, 2015.

[250] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[251] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066. Proceedings of Machine Learning Research, 2013.

[252] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[253] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31, 2018.

[254] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

[255] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[256] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[257] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[258] Tiago Santos and Roman Kern. A literature survey of early time series classification and deep learning. *SAMI@iKNOW*, 2016.

[259] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[260] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, 54(3):1–33, 2021.

[261] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: a survey. *arXiv preprint arXiv:2002.12478*, 2020.

[262] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. Proceedings of Machine Learning Research, 2013.

[263] Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018.

[264] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. Patient subtyping via time-aware LSTM networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 65–74. ACM, 2017.

[265] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: a machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 28, 2015.

[266] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Conference on Computer Vision and Pattern Recognition*, pages 7132–7141. IEEE, 2018.

[267] Jingkun Gao, Xiaomin Song, Qingsong Wen, Pichao Wang, Liang Sun, and Huan Xu. RobustTAD: Robust time series anomaly detection via decomposition and convolutional neural networks. *arXiv preprint arXiv:2002.09545*, 2020.

[268] Jordan Yeomans, Simon Thwaites, William SP Robertson, David Booth, Brian Ng, and Dominic Thewlis. Simulating time-series data for improved deep neural network performance. *IEEE Access*, 7:131248–131255, 2019.

[269] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[270] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[271] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[272] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[273] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. Proceedings of Machine Learning Research, 2023.

[274] Carson Eisenach, Yagna Patel, and Dhruv Madeka. MQTransformer: multi-horizon forecasts with context dependent and feedback-aware attention. *arXiv preprint arXiv:2009.14799*, 2020.

[275] Xingyu Wang, Hui Liu, Junzhao Du, Zhihan Yang, and Xiyao Dong. CLformer: locally grouped auto-correlation and convolutional transformer for long-term multivariate time series forecasting. *Engineering Applications of Artificial Intelligence*, 121:106042, 2023.

[276] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*, 2021.

[277] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[278] Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen. Voice2Series: reprogramming acoustic models for time series classification. In *International Conference on Machine Learning*, pages 11808–11819. Proceedings of Machine Learning Research, 2021.

[279] Jina Kim, Hyeongwon Kang, and Pilsung Kang. Time-series anomaly detection with stacked transformer representations and 1d convolutional network. *Engineering Applications of Artificial Intelligence*, 120:105964, 2023.

[280] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2114–2124. ACM, 2021.

[281] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: a benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[282] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: the efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[283] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[284] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

[285] Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *European Physical Journal C*, 79(7):1–11, 2019.

[286] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[287] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[288] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[289] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835*, 2018.

[290] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445. Association for the Advancement of Artificial Intelligence, 2020.

[291] Keyulu Xu, Mozhi Zhang, Stefanie Jegelka, and Kenji Kawaguchi. Optimization of graph neural networks: implicit acceleration by skip connections and more depth. In *International Conference on Machine Learning*, pages 11592–11602. Proceedings of Machine Learning Research, 2021.

[292] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[293] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[294] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[295] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: a review of methods and applications. *AI Open*, 1:57–81, 2020.

[296] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: a survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.

[297] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: a strong baseline. In *International Joint Conference on Neural Networks*, pages 1578–1585. IEEE, 2017.

[298] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749–760, 2018.

[299] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision*, pages 618–626, 2017.

[300] Baptiste Lafabregue, Jonathan Weber, Pierre Gançarski, and Germain Forestier. End-to-end deep representation learning for time series clustering: a comparative study. *Data Mining and Knowledge Discovery*, 36(1):29–81, 2022.

[301] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, 2019.

[302] Muhammad S Battikh and Artem A Lenskiy. Latent-insensitive autoencoders for anomaly detection. *Mathematics*, 10(1):112, 2021.

[303] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

[304] Sergio Naval Marimont and Giacomo Tarroni. Anomaly detection through latent space restoration using vector quantized variational autoencoders. In *International Symposium on Biomedical Imaging*, pages 1764–1767. IEEE, 2021.

[305] Oyebade K Oyedotun and Djamila Aouada. A closer look at autoencoders for unsupervised anomaly detection. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3793–3797. IEEE, 2022.

[306] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[307] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2828–2837. ACM, 2019.

[308] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *arXiv preprint arXiv:1406.2661*, 2014.

[309] Farzan Farnia and Asuman Ozdaglar. Do GANs always have Nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. Proceedings of Machine Learning Research, 2020.

[310] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[311] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. Proceedings of Machine Learning Research, 2017.

[312] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[313] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

[314] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: an overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[315] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. Hierarchical deep generative models for multi-rate multivariate time series. In *International Conference on Machine Learning*, pages 784–793. Proceedings of Machine Learning Research, 2018.

[316] Konstantinos Nikolaidis, Stein Kristiansen, Vera Goebel, Thomas Plagemann, Knut Liestøl, and Mohan Kankanhalli. Augmenting physiological time series data: a case study for sleep apnea detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 376–399. Springer, 2019.

[317] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv:1706.02633*, 2017.

[318] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[319] Jiaao Chen and Diyi Yang. Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization. *arXiv preprint arXiv:2010.01672*, 2020.

[320] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[321] Wei Liu, Sihan Chen, Longteng Guo, Xinxin Zhu, and Jing Liu. CPTR: full transformer network for image captioning. *arXiv preprint arXiv:2101.10804*, 2021.

[322] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[323] Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, et al. AlexaTM 20b: few-shot learning using a large-scale multilingual seq2seq model. *arXiv preprint arXiv:2208.01448*, 2022.

[324] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[325] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[326] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. Explainable artificial intelligence (XAI) on time series data: a survey. *arXiv preprint arXiv:2104.00950*, 2021.

[327] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[328] Zachary C Lipton. The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[329] Cynthia Rudin. Stop explaining black-box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[330] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[331] Nazanin Fouladgar, Marjan Alirezaie, and Kary Främling. Metrics and evaluations of time series explanations: an application in affect computing. *IEEE Access*, 10:23995–24009, 2022.

[332] Fatemeh Alizadeh, Margarita Esau, Gunnar Stevens, and Lena Cassens. eXplainable AI: take one step back, move two steps forward. *Mensch und Computer*, 2020.

[333] Advait Sarkar. Is explainable AI a race against model complexity? *arXiv preprint arXiv:2205.10119*, 2022.

[334] Devinder Kumar, Graham W Taylor, and Alexander Wong. Opening the black box of financial AI with clear-trade: a class-enhanced attentive response approach for explaining and visualizing deep learning-driven stock market prediction. *arXiv preprint arXiv:1709.01574*, 2017.

[335] Shoaib Ahmed Siddiqui, Dominique Mercier, Mohsin Munir, Andreas Dengel, and Sheraz Ahmed. TSViz: demystification of deep learning models for time-series analysis. *IEEE Access*, 7:67027–67040, 2019.

[336] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using SHAP. *arXiv preprint arXiv:1903.02407*, 2019.

[337] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[338] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

[339] Nantian Huang, Guobo Lu, and Dianguo Xu. A permutation importance-based feature selection method for short-term electricity load forecasting using random forest. *Energies*, 9(10):767, 2016.

[340] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. Proceedings of Machine Learning Research, 2017.

[341] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[342] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International Conference on Machine Learning*, pages 9269–9278. Proceedings of Machine Learning Research, 2020.

[343] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: an approach to evaluating interpretability of machine learning. *arXiv preprint arXiv:1806.00069*, page 118, 2018.

[344] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. On top-k recommendation using social networks. In *Proceedings of the ACM Conference on Recommender Systems*, pages 67–74. ACM, 2012.

[345] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 95–104. ACM, 2018.

[346] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[347] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[348] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

[349] Marco Maggipinto, Alessandro Beghi, and Gian Antonio Susto. A deep learning-based approach to anomaly detection with 2-dimensional data in manufacturing. In *International Conference on Industrial Informatics*, volume 1, pages 187–192. IEEE, 2019.

[350] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.

[351] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[352] Gavneet Singh Chadha, Arfyan Rabbani, and Andreas Schwung. Comparison of semi-supervised deep neural networks for anomaly detection in industrial processes. In *International Conference on Industrial Informatics*, volume 1, pages 214–219. IEEE, 2019.

[353] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[354] Valdas Rapsevicius, CMS DQM Group, et al. CMS run registry: data certification bookkeeping and publication system. In *Journal of Physics: Conference Series*, volume 331, page 042038. IOP Publishing, 2011.

[355] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Proceedings of Computer Vision and Pattern Recognition*, pages 2528–2535. IEEE, 2010.

[356] Leslie N Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pages 369–386. SPIE, 2019.

[357] Grace Cummings and The CMS-HCAL Collaboration. CMS HCAL VTRx-induced communication loss and mitigation. CERN private communications, 2021.

[358] Yangdong He and Jiabao Zhao. Temporal convolutional networks for anomaly detection in time series. In *Journal of Physics: Conference Series*, volume 1213, page 042050. IOP Publishing, 2019.

[359] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: a simple approach to mitigating KL vanishing. *arXiv:1903.10145*, 2019.

[360] Ingvar Eide and Frank Westad. Automated multivariate analysis of multi-sensor data submitted online: real-time environmental monitoring. *PLoS One*, 13(1):e0189443, 2018.

[361] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: an overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

[362] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: theory and applications*. Springer Science and Business Media, 2005.

[363] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise Reduction in Speech Processing*, pages 1–4, 2009.

[364] Jorge Martinez-Gil, Georg Buchgeher, David Gabauer, Bernhard Freudenthaler, Dominik Filipiak, and Anna Fensel. Root cause analysis in the industrial domain using knowledge graphs: a case study on power transformers. *Procedia Computer Science*, 200:944–953, 2022.

[365] Kristof Böhmer and Stefanie Rinderle-Ma. Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users. *Information Systems*, 90:101438, 2020.

[366] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. STL: a seasonal-trend decomposition procedure based on Loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[367] Tom Puech, Matthieu Boussard, Anthony D'Amato, and Gaëtan Millerand. A fully automated periodicity detection in time series. In *Advanced Analytics and Learning on Temporal Data: ECML PKDD Workshop*, pages 43–54. Springer, 2020.

[368] Arik Ermshaus, Patrick Schäfer, and Ulf Leser. ClaSP: parameter-free time series segmentation. *Data Mining and Knowledge Discovery*, 37(3):1262–1300, 2023.

[369] Kasun Bandara, Rob J Hyndman, and Christoph Bergmeir. MSTL: a seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *arXiv preprint arXiv:2107.13462*, 2021.

[370] Robert W Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V: Proceedings of the 5th Australian Conference*, pages 28–43. Springer, 1977.

[371] Yvonne M Bishop, Stephen E Fienberg, and Paul W Holland. *Discrete multivariate analysis: theory and practice.* Springer Science and Business Media, 2007.

[372] Kevin Patrick Murphy. *Dynamic Bayesian networks: representation, inference and learning.* University of California, Berkeley, 2002.

[373] Ali Sheidaei, Abbas Rahimi Foroushani, Kimiya Gohari, and Hojjat Zeraati. A novel dynamic Bayesian network approach for data mining and survival data analysis. *Medical Informatics and Decision Making*, 22(1):1–15, 2022.

[374] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[375] Alain Celisse, Guillemette Marot, Morgane Pierre-Jean, and GJ Rigaill. New efficient algorithms for multiple change-point detection with reproducing kernels. *Computational Statistics and Data Analysis*, 128:200–220, 2018.

[376] Jonas Peters and Peter Bühlmann. Structural intervention distance for evaluating causal graphs. *Neural Computation*, 27(3):771–799, 2015.

[377] Judea Pearl et al. Models, reasoning and inference. *Cambridge University Press*, 19(2):3, 2000.

[378] Ilya Shpitser, Tyler VanderWeele, and James M Robins. On the validity of covariate adjustment for estimating causal effects. *arXiv preprint arXiv:1203.3515*, 2012.

[379] Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gcastle: a python toolbox for causal discovery. *arXiv preprint arXiv:2111.15155*, 2021.

[380] Alexandra M Carvalho. Scoring functions for learning Bayesian networks. *Inesc-id Tec. Rep*, 12:1–48, 2009.

[381] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, Kenneth Bollen, and Patrik Hoyer. Directlingam: a direct method for learning a linear non-Gaussian structural equation model. *Journal of Machine Learning Research*, 12:1225–1248, 2011.

[382] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

[383] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

[384] Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33:17943–17954, 2020.

[385] Ignavier Ng, Shengyu Zhu, Zhitang Chen, and Zhuangyan Fang. A graph autoencoder approach to causal structure learning. *arXiv preprint arXiv:1911.07420*, 2019.

[386] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*, 2019.

[387] Xiaoqiang Wang, Yali Du, Shengyu Zhu, Liangjun Ke, Zhitang Chen, Jianye Hao, and Jun Wang. Ordering-based causal discovery with reinforcement learning. *arXiv preprint arXiv:2105.06631*, 2021.

[388] Massimiliano Galli, Enric Tejedor, and Stefan Wunsch. A new pyroot: Modern, interoperable and more pythonic. In *European Physical Journal Web of Conferences*, volume 245, page 06004. EDP Sciences, 2020.

[389] Azul Garza and Max Mergenthaler-Canseco. TimeGPT-1. *arXiv preprint arXiv:2310.03589*, 2023.