Advances in Deep Learning Towards Fire Emergency Application: Novel Architectures, Techniques and Applications of Neural Networks

Jivitesh Sharma

Doctoral Dissertations at the University of Agder 308

Jivitesh Sharma

Advances in Deep Learning Towards Fire Emergency Application: Novel Architectures, Techniques and Applications of Neural Networks

Doctoral Dissertation for the Degree *Philosophiae Doctor (PhD)* at
the Faculty of Engineering and Science,
Department of Information and Communication Technology,
Specialization in Artificial Intelligence

University of Agder
Faculty of Engineering and Science
2020

Doctoral Dissertations at the University of Agder 308
ISBN: 978-82-8427-011-1
ISSN: 1504-9272

Printed by 07 Media
Kristiansand

Dedicated to my parents

Mukul Sharma and Vandana Sharma

And my loving wife

Charul Giri

Abstract

Deep Learning has been successfully used in various applications, and recently, there has been an increasing interest in applying deep learning in emergency management. However, there are still many significant challenges that limit the use of deep learning in the latter application domain. In this thesis, we address some of these challenges and propose novel deep learning methods and architectures.

The challenges we address fall in these three areas of emergency management: *Detection* of the emergency (fire), *Analysis* of the situation without human intervention and finally *Evacuation* Planning. In this thesis, we have used computer vision tasks of image classification and semantic segmentation, as well as sound recognition, for detection and analysis. For evacuation planning, we have used deep reinforcement learning.

The detection phase involves detecting whether there is a fire emergency or not. Previous methods proposed for the detection problem have been prone to overfitting, large inference times and requiring tremendous amounts of training data. To overcome these issues, we propose to use state-of-the-art CNNs with pre-trained weights. These are trained to distinguish between fire and normal images, by fine-tuning their parameters on our own custom dataset. To further reduce inference time and reduce required training time, we also propose a CNN-ELM hybrid model. Finally, we propose a more general-purpose emergency detection method using audio signals. For this, we use multiple features extracted by signal processing methods, proposing a novel attention-based deep separable CNN architecture.

We next propose the first deep learning based semantic segmentation approach for visual analysis of the emergency environment, in order to provide adequate visual information an emergency situation. A deep CNN is used to parse the fire emergency scene and segment most types of objects, including segmentation of people and the fire itself. The objects are segmented according to their build material and their vulnerability to catch fire, which can give a rough estimate of fire spread. Since people (and other living creatures) are also segmented. We train the models with multitask learning, so that the encoder/backbone of the segmentation model can be used to classify fire emergencies and activate the decoder if needed.

This reduces unnecessary computation.

Evacuation planning involves coming up with an optimal policy to evacuate as many people as possible in a timely manner. There has been some research in this area, but no one has been able to tackle the full complexity of the fire evacuation problem. Most works have been limited to either human behaviour modelling, simple maze environment based path planning or fire spread modelling. We encompass most of the aspects of a fire emergency situation and address the issue in a wholesome manner. A deep reinforcement learning strategy is proposed to obtain a near optimal evacuation plan for the whole environment. A novel method involving transfer learning, tabular Q-learning and Deep Q-Networks is used to learn an evacuation plan which can evacuate all people inside a building in the smallest number of steps. In order to train this reinforcement learning method, a novel environment for fire evacuation is built, using the OpenAI gym framework. The graph based fire evacuation environment consists of realistic features such as bottlenecks, fire spread and an exponential decaying reward function to force the reinforcement learning agent to learn the shortest evacuation paths.

The proposed models are disjoint, but can be easily connected to each other to pass useful information. Overall, our methods show excellent results on simulation and/or open sourced data. We therefore believe that they form a promising toolbox for further exploration in real world emergency management systems. The primary focus of this thesis has thus been to advance the state-of-the-art for AI models applied to emergency management or set a new paradigm for problems that have not been fully addressed by AI before.

Preface

This dissertation is a result of the research work carried out at the Department of Information and Communication Technology (ICT), University of Agder (UiA), Grimstad, Norway, from January 2017 to May 2020. During my Ph.D. study, my main supervisor has been Professor Ole-Christoffer Granmo, University of Agder, and my co-supervisor has been Associate Professor Morten Goodwin, University of Agder (UiA), Norway.

I have also worked on the GHO-DL Industrial project with Dr. Bernt-Viggo Matheussen from Agder Energi AS. The total time spent on this project was 1 year, where I worked on the project in three phases of 4 months each, between January 2018 and June 2020. This project has been supported by Agder Energi AS, the Norwegian Research Council (ENERGIX program), and the University of Agder.

Production note: LaTeX has been adopted as the tool for writing this dissertation, as well as the papers produced during my Ph.D. study. The mathematical calculations and simulation results are obtained by using PYTHON and supporting AI libraries.

Acknowledgments

First of all, I would like to express my intense gratitude to my supervisor, Professor Ole-Christoffer Granmo. Without his assistance and guidance, this endeavor would not have been possible. His encouragement, enthusiasm, and vision always motivated me to do fruitful research. I am deeply indebted to him for his meticulous reviews of my manuscripts. His comments and feedback always helped to ensure the manuscripts with high quality. His contribution in my personal and professional development is enormous. During the ups and downs of the journey, he is the only person who always came with a solution and showed the next logical step.

I am grateful to my co-supervisor Professor Morten Goodwin for his support and guidance in my research. His insightful comments and feedback always helped to improve the quality of the manuscripts. He has been and still is one of my mentors in Deep Learning and Computer Vision. He is an expert in AI, whom I relied on during my time as a PhD student. Both my supervisors have not only played the role of advisors but also have been guardians.

I would like to extend my thanks to my co-authors, Jahn Thomas Fidje and Per-Arne Andersen for their cooperation and contributions. I wish to thank the Ph.D. coordinators at the Faculty of Engineering and Science, UiA, Kristine Evensen Reinfjord, Tonje Sti and Emma Elisabeth Horneman for their administrative support. I also would like to thank all the professors at the Department of ICT who always encouraged and motivated me.

I am also grateful to my teammates in CAIR (Center for Artificial Intelligence Research) for their feedback, help, and encouragements through these years. I am thankful to my old and new office-mates Dr. Vimala Nunavath, Ayan Chatterjee, Micheal Dutt, Darshana Kuruge Abeyrathna and Pankaj Khatiwada for their cooperation. I really enjoyed the technical as well as non-technical discussions that we had in the office. Many thanks to Mohamed Gafar Ahmed Elnourani, Luis Miguel Lopez-Ramos, Martin Holen, Saeed Gorji, Bimal Bhattarai, Rohan Yadav for their support throughout the Ph.D. journey.

Although far away from my home country, India, where my family live, the love, encouragement, and support from my parents, Mukul Sharma and Vandana

Sharma. Taking this opportunity, I express my eternal gratitude to them for their blessings, love, understanding, and support. Their confidence on me is the original source that leads me to every tiny success in my life. I also take great pride in thanking my wife, Charul Giri, who always supported, motivated, and encouraged me in this endeavor. I believe my late grandfather and grandmother would have been really proud of me.

Many thanks to all of you for your encouragement!

Jivitesh Sharma
June 2020
Grimstad, Norway

List of Publications

The author of this dissertation is the first author and the principal contributor of all the included papers listed below. Papers A-F in the first set of the following list are selected to represent the main research achievements and are reproduced as Part II of this dissertation. Other papers which are listed in the second set are some other contributions towards Artificial Intelligence Research.

Papers Included in the Dissertation

Paper A Sharma, J., Granmo, O.C., Goodwin, M., and Fidje, J. T., "Deep Convolutional Neural Networks for Fire Detection in Images.", In *In International Conference on Engineering Applications of Neural Networks (EANN)*, pp. 183-193. Springer, Aug. 2017.

Paper B Sharma, J., Granmo, O.C., and Goodwin, M., "Deep CNN-ELM Hybrid Models for Fire Detection in Images.", In *In International Conference on Artificial Neural Networks (ICANN)*, pp. 245-259. Springer, Oct. 2018.

Paper C Sharma, J., Andersen, P. A., Granmo, O. C., and Goodwin, M., "Deep Q-Learning with Q-Matrix Transfer Learning for Novel Fire Evacuation Environment.", In *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1-19., Feb. 2020.

Paper D Sharma, J., Granmo, O. C., and Goodwin, M., "Emergency Analysis: Multitask Learning with Deep Convolutional Neural Networks for Fire Emergency Scene Parsing.", In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE).*, pages 1-15, Springer, July. 2021.

Paper E Sharma, J., Granmo, O. C., and Goodwin, M., "Environment Sound Classification using Multiple Feature Channels and Attention based Deep Convolutional Neural Network.", In *Conference of the International Speech Communication Association (INTERSPEECH).*, pages 1-7, Oct. 2020.

Paper F Sharma, J., Granmo, O. C., and Goodwin, M., "Emergency detection with Environment Sound using Deep Convolutional Neural Networks.", In *International Congress on Information and Communication Technology (ICICT).*, pages 1-10, Springer, Feb. 2020.

Other Publications

Paper 7 Matheussen, B. V., Granmo, O. C., and Sharma, J., "Hydropower Optimization Using Deep Learning.", In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE).*, pp. 110-122, Springer, July. 2019.

Paper 8 Sharma, J., Matheussen, B. V., Glimsdal, S., and Granmo, O. C., "Hydropower Optimization Using Split-Window, Meta-Heuristic and Genetic Algorithms.", In *IEEE International Conference On Machine Learning And Applications (ICMLA).*, pp. 882-888, Dec. 2019.

Paper 9 Sharma, J., Giri, C., Granmo, O.C. and Goodwin, M., "Multi-layer Intrusion Detection System with ExtraTrees feature selection, Extreme Learning Machine ensemble, and Softmax aggregation.", In *EURASIP Journal on Information Security.*, p.15, Springer, Dec. 2019.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
SVM	Support Vector Machines
DL	Deep Learning
NN	Neural Networks
DNN	Deep Neural Networks
CNN	Convolutional Neural Networks
DCNN	Deep Convolutional Neural Networks
BN	Batch Normalization
ReLU	Rectified Linear Unit
ResNet	Residual Network
ELM	Extreme Learning Machine
WELM	Weighted Extreme Learning Machine
BP	Back-propagation
SGD	Stochastic Gradient Descent
ADAM	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient
CE	Cross Entropy
BCE	Binary Cross Entropy
GAN	Generative Adversarial Networks
PSPNet	Pyramid Scene Parsing Network
FCN	Fully Convolutional Network
SegNet	Segmentation Network
MDP	Markov Decision Process
DQN	Deep Q-Networks
DDQN	Double Deep Q-Networks
VPG	Vanilla Policy Gradient
PPO	Proximal Policy Optimization
TRPO	Trust-Region Policy Optimization

A2C	Advantage Actor Critic
A3C	Asynchronous Advantage Actor Critic
SARSA	State Action Reward State Action
ACKTR	Actor Critic with Kronecker Factored Trust-Region
DQfD	Deep Q-learning from Demonstrations
DDPG	Deep Deterministic Policy Gradient
TTQL	Task Transfer Q-learning
MFCC	Mel-Frequency Cepstral Coefficients
GFCC	Gammatone Frequency Cepstral Coefficients
CQT	Constant Q-Transform
STFT	Short-time Fourier Transform
ESC	Environment Sound Classification

PART I

Chapter 1

Introduction

In this chapter, we first give a brief introduction of Emergency Management and its stages, in relation to this thesis. Then the importance of using Deep Learning in these stages is highlighted. The motivation of this Ph.D. dissertation is discussed along with the overview of the research questions. Furthermore, the goals and approaches are explained and the structure of the dissertation is also outlined.

1.1 Emergency Management

There have been countless natural and man-made disasters that have caused irreversible life and property damage to innumerable people across the globe. According to [2], in 2017 alone, more than 11,000 people lost their lives or went missing during disasters, while millions were left homeless, worldwide. In the same year, economic losses due to natural and man-made disasters amounted to 337 billion USD. In 2017, there were 301 natural or man-made disasters worldwide [2]. To prevent or minimize human and economic losses as much as possible, emergency management departments has been put in place in almost every country and large organization. The importance of emergency management is well understood in the civilized world.

Emergency Management is organized analysis, planning, decision making, and assignment of available resources to mitigate, prepare for, respond to, and recover from the effects of all hazards. The importance of emergency management is well established. There exists a plethora of emergency management systems, but only a few leverage deep learning. Even though deep learning has been successfully used in many different applications, however current deep learning methods suffer from drawbacks that limit their application in the area of emergency management. So, in this thesis, we focus on building an AI decision support system for each step of the emergency. There are several emergency situations that can arise anywhere in the world including floods, droughts, avalanches, tsunamis, hurricanes, earthquakes, volcano eruptions, terrorism, pandemics etc. This thesis focuses on addressing limitations of deep learning approaches motivated by fire emergencies.

1.2 Motivation and Research Questions

A detailed and comprehensive survey of artificial intelligence based techniques used for emergency management problems can be found in [3]. It enlists various AI based methods used during different phases of the emergency management procedure. There have been various contributions in AI that have led to advances in AI based emergency management methods. For a detailed overview of these methods, please refer to sections 3.1.1, 3.1.2, 3.3. However, these methods either suffer from various shortcomings that hinder performance such as slow training, inference time and overfitting or fail to focus on some key issues like lack of visual analysis and reinforcement learning based full-scale evacuation planning. We enlist the research questions that we answer in this thesis, the motivation behind those questions and our approach. In doing so, we aim to advance the state-of-the-art for AI methods on some specific emergency management applications.

Question 1: Can CNNs be efficiently used for detecting fire while reducing overfitting, and can transfer learning be used to further improve performance?

Motivation: Fire is a highly unlikely scenario in the modern world. So, it must be treated like one by having training data that is highly imbalanced. Shallower CNN models are prone to overfitting on imbalanced class distribution. While being a powerful technique, transfer learning has had limited attention in fire detection.

Approach: To address this limitation, we propose to employ state-of-the-art CNN models like VGG16 and ResNet50 for the task of fire detection. We use transfer learning to further improve performance. To evaluate the models on imbalanced data, we create our own dataset with a fire to non-fire images ratio of 1:9.

Question 2: Since state-of-the-art CNNs are quite slow, can we use a novel hybrid model to boost performance as well as reduce inference time?

Motivation: Emergency detection is an application in which inference times matter a lot. Detection an emergency must be performed in the shortest amount of time with maximum precision. This motivates us to build a model that can achieve lower inference times while maintaining accurate detection rates of state-of-the-art models.

Approach: We address this issue by proposing a hybrid model consisting of state-of-the-art CNN feature extraction and an ELM (Extreme Learning Machine) classifier, replacing the fully connected layers and the softmax classifier. Most of the operational complexity of CNN models lies in the fully connected layers and backpropagation through it. The ELM is a one-shot learning algorithm which does not require backpropagation that makes it incredibly fast to train. And, since it has a single layer with lower number of neurons, it has a much lower inference time.

Question 3: Can reinforcement learning be used for evacuation planning in a highly realistic, dynamic environment in a timeliness manner?

Motivation: Most of the research on evacuation planning has been insufficient in terms of scale and complexity. There is no standard benchmark environment to test different approaches. Also, current state-of-the-art RL methods might fail in a highly dynamic environment with a large state and action space.

Approach: We propose a novel RL method to train agents to evacuate people in the least number of time-steps. We also propose the first graph-based evacuation environment to train RL agents. The environment consists of realistic features such as fire spread, bottlenecks, uncertainty and full action connectivity. A standard RL method cannot be trained directly on such as complex environment. So, we first pretrain a DQN agent to learn the shortest path from each node in building graph to the nearest exit. This information is useful during evacuation planning. Then, the pretrained agent is trained on the full fire evacuation environment. In order to scale the technique to real world scenarios, we propose an action importance based attention mechanism that is able to reduce the action space to a manageable size.

Question 4: Can semantic segmentation be used to detect objects that are vulnerable to catch fire and segment other relevant entities to analyse the emergency environment?

Motivation: There has been a lot of research in pre-emergency analysis. However, analysis is extremely important during an emergency as well. Visual analysis using computer vision could provide essential information without human intervention.

Approach: We propose to use semantic segmentation for fire emergency scene parsing. We propose to segment objects based on their build material such as wood, plastic, cloth etc. along with segmenting fire, smoke and people (to get a rough head count). To the best of our knowledge, this is the first visual analysis tool for fire emergencies. Since this is the first visual analysis method, we also create our own fire scene parsing dataset. In order to reduce unnecessary computation during inference, we use multitask learning so that the encoder/backbone learns to classify between fire and normal images and the decoder is activated only if a fire is detected.

Question 5: Can we further improve environment sound classification by using multiple feature channels, separable convolutions and an attention mechanism?

Motivation: Current state-of-the-art environment sound classification models suffer from complexity and not being able to fully extract relevant features from audio signals. Also, since time and frequency domain features represent different kinds of information, it might be beneficial to treat them separately.

Approach: To address these limitations, we propose a novel model for environment sound classification. We propose to use multiple feature extraction techniques like Mel-Frequency Cepstral Coefficients (MFCC), Gammatone Frequency Cepstral Coefficients (GFCC), the Constant Q-transform (CQT) and Chromagram to extract complementary information. This is used as a multiple channel input to the CNN classifier. Our CNN consists of spatially separable convolution and pooling layers to process time and frequency domain separately. To further improve performance and keep the parameter count as low as possible, we use depthwise separable convolutions in a parallel attention block.

Question 6: Can an environment sound classification approach be used as a general purpose audio based emergency detection system?

Motivation: Sometimes, visual detection can be prone to false alarms and inaccuracies. A sound based detection method can be used to complement the visual detection model. Some information is most readily available visually (e.g., the extent of a fire), while other information is perhaps best captured through sound (e.g., screaming). So, it might be beneficial to use both approaches.

Approach: We use our state-of-the-art environment sound classification model, for this task. We make a few changes to the model to reduce the number of parameters. Since this is a binary classification problem, model complexity can be reduced. This is done by reducing the number of feature maps in the convolution layers and removing the attention block. The signal feature extraction part is kept the same with multiple feature channel input to the CNN classifier.

Limitations: In this thesis, the focus is not to build a complete emergency management system or evaluate it in real-life use. Instead, we seek to advance deep learning techniques by addressing fundamental weaknesses in current state-of-the-art methods. The main focus of the thesis lies in addressing issues in AI based methods used for detection, analysis and evacuation phases of the emergency management procedure, in a disjoint manner. The scope of this thesis is limited to the proposal of novel deep learning techniques, architectures and procedures applied to parts of emergency management. The methods proposed in this thesis haven't been tested in the real-world, and we do not claim to reciprocate the simulation results on real-life examples.
Overall, the whole system is disjoint and versatile and can be adapted for other types of emergency situations as well. Also, information sharing between different (disjoint) modules is also straightforward and can be implemented easily as and when required. Hence, we propose and employ novel AI methods, architectures and learning techniques to outperform existing AI based solutions or set a new paradigm of AI methods with potential application to different parts of the emergency management process.

1.3 Publications

We solve each of the three stages of the fire emergency management process using deep learning and neural networks. We employ various areas of AI to solve the problems of detection, analysis and evacuation. We list the contributions of this thesis below; each contribution is described in detail in Chapter III and the associated papers published are presented in Part II of the thesis. Here, we present a short summary of our papers:

Paper A We discovered that a traditional CNN performs relatively poorly when evaluated on the more realistically balanced benchmark dataset provided in this paper. We therefore propose to use even deeper Convolutional Neural Networks for fire detection in images, and enhancing these with fine tuning based on a fully connected layer. We use two pretrained state-of-the-art Deep CNNs, VGG16 and Resnet50, to develop our fire detection system. The Deep CNNs are tested on our imbalanced dataset, which we have assembled to replicate real world scenarios. It includes images that are particularly difficult to classify and that are deliberately unbalanced by including significantly more non-fire images than fire images. The dataset has been made available online. Our results show that adding fully connected layers for fine tuning indeed does increase accuracy, however, this also increases training time. Overall, we found that our deeper CNNs give good performance on a more challenging dataset, with Resnet50 slightly outperforming VGG16. These results may thus lead to more successful fire detection systems in practice. This paper addresses Research Question 1.

Paper B In this paper, we propose a hybrid model consisting of a Deep Convolutional feature extractor followed by a fast and accurate classifier, the Extreme Learning Machine, for the purpose of fire detection in images. The reason behind using such a model is that Deep CNNs used for image classification take a very long time to train. Even with pre-trained models, the fully connected layers need to be trained with backpropagation, which can be very slow. In contrast, we propose to employ the Extreme Learning Machine (ELM) as the final classifier trained on pre-trained Deep CNN feature extractor. We apply this hybrid model on the problem of fire detection in images. We use state of the art Deep CNNs: VGG16 and Resnet50 and replace the softmax classifier with the ELM classifier. For both the VGG16 and Resnet50, the number of fully connected layers is also reduced. Especially in VGG16, which has 3 fully connected layers of 4096 neurons each followed by a softmax classifier, we replace two of these with an ELM classifier. The difference in convergence rate between fine-tuning the fully connected layers of pre-trained models and training an ELM classifier are enormous, around 20x to 51x speed-up. Also, we show that using an ELM classifier increases the accuracy of the system by 2.8% to 7.1% depending on the CNN feature extractor. We also compare our hybrid architecture with another hybrid architecture, i.e. the CNN-SVM model. Using SVM as the classifier does improve accuracy compared to state-of-the-art deep CNNs. But our Deep CNN-ELM model is able to outperform the Deep CNN-SVM models. This paper addresses Research Question 2.

Paper C In this paper, we propose the first fire evacuation environment to train reinforcement learning agents for evacuation planning. The environment is modelled as a graph capturing the building structure. It consists of realistic features like fire spread, uncertainty and bottlenecks. We have implemented the environment in the OpenAI gym format, to facilitate future research. We also propose a new reinforcement learning approach that entails pretraining the network weights of a DQN based agent (DQN/Double-DQN/Dueling-DQN) to incorporate information on the shortest path to the exit. We achieved this by using tabular Q-learning to learn the shortest path on the building model's graph. This information is transferred to the network by deliberately overfitting it on the Q-matrix. Then, the pretrained DQN model is trained on the fire evacuation environment to generate the optimal evacuation path under time varying conditions due to fire spread, bottlenecks and uncertainty. We perform comparisons of the proposed approach with state-of-the-art reinforcement learning algorithms like DQN, DDQN, Dueling-DQN, PPO, VPG, SARSA, A2C and ACKTR. The results show that our method is able to outperform state-of-the-art models by a huge margin including the original DQN based models. Finally, we test our model on a large and complex real building consisting of 91 rooms, with the possibility to move to any other room, hence giving 8281 actions. In order to reduce the action space, we propose a strategy that involves one step simulation. That is, an action importance vector is added to the final output of the pretrained DQN and acts like an attention mechanism. Using this strategy, the action space is reduced by 90.1%. In this manner, we are able to deal with large action spaces. Hence, our model achieves near optimal performance on the real world emergency environment. This paper addresses Research Question 3.

Paper D In this paper, we introduce a novel application of using scene/ semantic image segmentation for emergency situation analysis. During an emergency state, it is imperative to analyse the situation thoroughly before planning a response. In this work, we analyse fire emergency situations. To analyse a fire emergency scene, we propose to use deep convolutional image segmentation networks to identify and classify objects in a scene based on their build material and their vulnerability to catch fire. We also segment people and other living beings and fire, obviously. We introduce our own fire emergency scene segmentation dataset for this purpose. It consists of real world images with objects annotated on the basis of their build material. Including people (and any other living beings) and fire, the dataset consists of 10 segmentation classes. For this task, we propose to use the advanced computer vision technique of semantic image segmentation. We use state-of-the-art segmentation models: DeepLabv3, DeepLabv3+, PSPNet, FCN, SegNet and UNet to compare and evaluate their performance on the fire emergency scene parsing task. During inference time, we only run the encoder (backbone) network to determine whether there is a fire or not in the image. If there is a fire, only then the decoder is activated to segment the emergency scene. This results in dispensing with unnecessary computation, i.e. the decoder. We achieve this by using multitask learning. We also experiment with fine-tuning pretrained models versus training models from scratch. We show the importance of transfer learning and the difference in performance between models pretrained on different benchmark datasets. The results show that segmentation models can accurately analyse an emergency situation, if properly trained to do so. Our fire emergency scene parsing dataset is available here: https://github.com/cair.. This paper addresses Research Question 4.

Paper E In this paper, we propose a model for the Environment Sound Classification Task (ESC) that consists of multiple feature channels given as input to a Deep Convolutional Neural Network (CNN) with Attention mechanism. The novelty of the paper lies in using multiple feature channels consisting of Mel-Frequency Cepstral Coefficients (MFCC), Gammatone Frequency Cepstral Coefficients (GFCC), the Constant Q-transform (CQT) and Chromagram. And, we employ a deeper CNN (DCNN) compared to previous models, consisting of spatially separable convolutions working on time and feature domain separately. Alongside, we use attention modules that perform channel and spatial attention together. We use the mix-up data augmentation technique to further boost performance. Our model is able to achieve state-of-the-art performance on three benchmark environment sound classification datasets, i.e. the UrbanSound8K (97.52%), ESC-10 (94.75%) and ESC-50 (87.45%). This paper addresses Research Question 5.

Paper F In this paper we propose a generic emergency detection system using only the sound produced in the environment. For this task, we employ multiple audio feature extraction techniques like the Mel-Frequency Cepstral Coefficients, Gammatone Frequency Cepstral Coefficients, Constant Q-transform and Chromagram. After feature extraction, a Deep Convolutional Neural Network (CNN) is used to classify an audio signal as a potential emergency situation or not. The entire model is based on our previous work that set the new state-of-the-art in the Environment Sound Classification (ESC) task (Paper E). We combine the benchmark ESC datasets: UrbanSound8K and ESC-50 (ESC-10 is a subset of ESC-50), and reduce the problem to a binary classification problem. This is done by aggregating sound classes such as sirens, fire crackling, glass breaking, gun shot as the emergency class and others as normal. Even though there are only two classes to distinguish, they are highly imbalanced. To overcome this difficulty we introduce class weights in calculating the loss while training the model. Our model is able to achieve 99.56% emergency detection accuracy. This paper addresses Research Question 6.

Each of the contributions listed above tries to solve a part of the fire emergency management problem. A pictorial representation of the flow of the content of our contributions in accordance with the aim of this thesis is shown in Figure 1.1. Our papers address the problems in the three phases of the fire emergency hazard and provide a solution that could potentially build a coherent fire emergency management system.

As can be seen from Figure 1.1, the modules in each phase are interconnected to each other, but also act as disjoint modules to provide separate information to the emergency personnel. The red arrows indicate the information flow within the detection phase. In the same manner, the blue and green arrows show the flow within the analysis and evacuation phases respectively. The yellow arrows designate interconnectivity between the phases and the decisions produced by the detection phase. Note that, the analysis and evacuation phases initiate execution only if the detection phase detects an emergency. The detection phase continuously monitors the environment while the analysis and evacuation phases kick-in once an emergency is detected.

1.4 Thesis Outline

The dissertation is organized into two parts. Part I contains an overview of the work carried out throughout this Ph.D. study and Part II includes a collection of six published or submitted papers, which are mentioned in the list of publications. In addition to the introduction chapter presented above, the following chapters are included.

- Chapter II presents some background and preliminary information of various techniques and methods used in the thesis, as this thesis involves image classification, semantic segmentation, reinforcement learning and sound processing and classification.

- Chapter III explains the contributions of this thesis in detail. It is divided into three sections which are the three stages of fire emergency management. Each section describes our contributions to each of the three stages, i.e. Detection, Analysis and Evacuation. The motivation, intuition, novelty, methodology and results for each contribution are explained extensively.
- Chapter IV concludes the thesis and discusses the implications of the outcomes of the thesis. It also contains potential future research directions that can further improve on the work presented in the thesis. This chapter also concludes Part I of the thesis.
- In Part II of the thesis, all publications pertaining to the thesis are presented in their entirety. There are six publications labelled as Paper A to F. The papers are not listed in chronological order, instead they are listed in accordance with the flow of the thesis and the ordering of the fire emergency management stages.

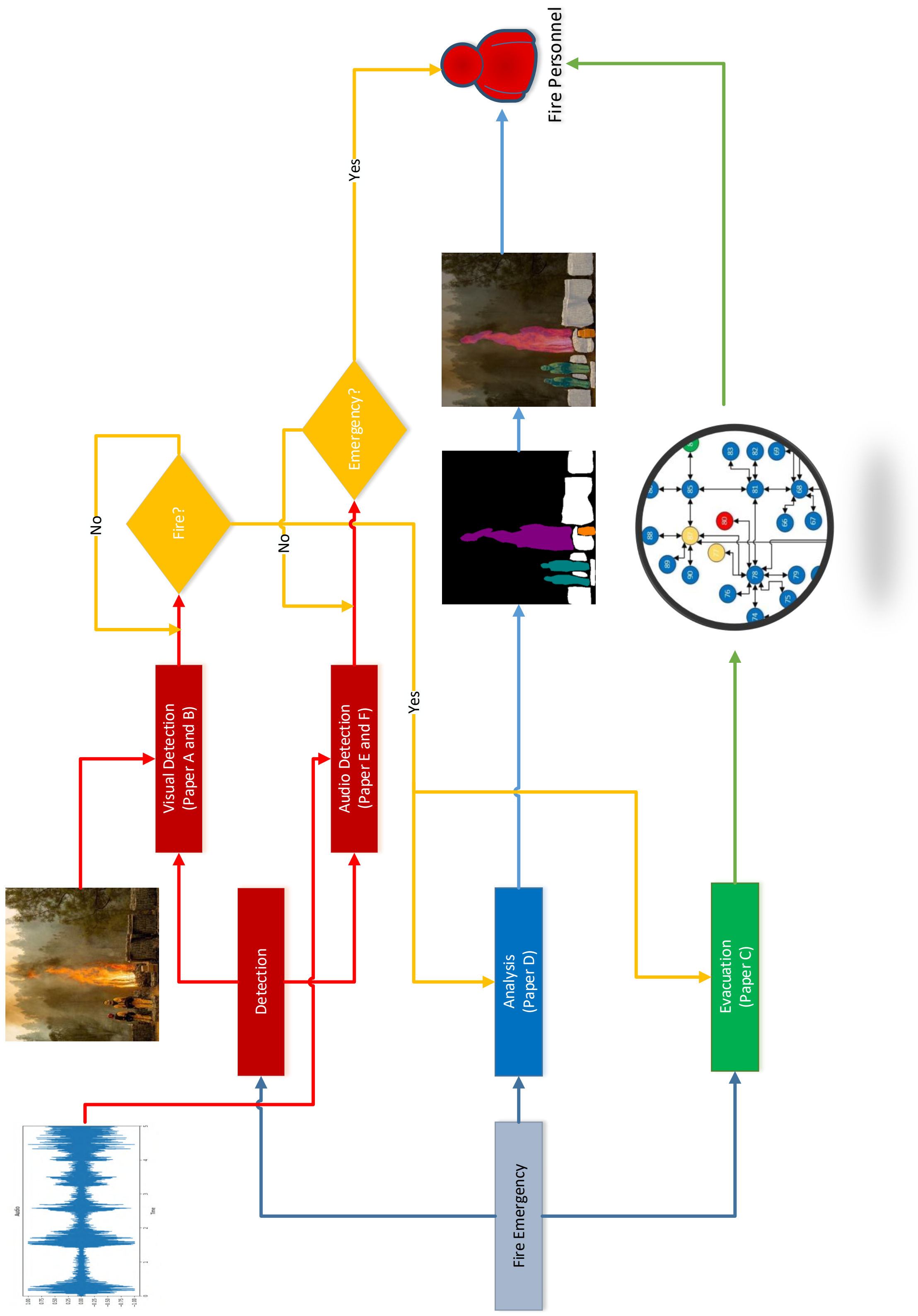


Figure 1.1: Organization of Contributions

Chapter 2

Background

In this chapter, we briefly describe the background and preliminary information needed to understand the thesis. We explain concepts that have been used throughout this thesis. First, Convolutional Neural Networks have been briefly introduced which have been extensively used in this thesis and are a recurring theme in the papers associated with this thesis. Next, we move on to signal processing. Some signal processing methods have been explained here that have been used in this thesis. We also include the Extreme Learning Machine algorithm that has been used to create hybrid image classification methods. Finally, we describe Reinforcement Learning and Q-learning methods that have been widely used in this thesis.

2.1 Convolutional Neural Networks

The Convolutional Neural Network was first introduced in 1980 by Kunihiko Fukushima, called Neocognitron [4]. The CNN is designed to take advantage of two dimensional structures like 2D Images and capture local spatial patterns. This is achieved with local connections and tied weights. It consists of one or more convolution layers with pooling layers between them, followed by one or more fully connected layers, as in a standard multilayer perceptron. CNNs are easier to train compared to Deep Neural Networks because they have fewer parameters and local receptive fields.
In CNNs, kernels/filters are used to see where particular features are present in an image by convolution with the image. The size of the filters gives rise to locally connected structure which are each convolved with the image to produce feature maps. The feature maps are usually subsampled using mean or max pooling. The reduction is parameters is due to the fact that convolution layers share weights. The reason behind parameter sharing is that we make an assumption, that the statistics

of a patch of a natural image are the same as any other patch of the image, which suggests that features learned at a location can also be learned for other locations. So, we can apply this learned feature detector anywhere in the image. This makes CNNs ideal feature extractors for images. Deeper CNNs with many layers have been used for various applications especially image classification where they perform much better than any other technique. Having more layers enable a larger receptive field which in turn enables the model to capture higher level abstractions. The CNNs models used in this thesis comprise of repetitions of Convolution layers followed by Batch Normalization and finally ReLU as activation function. We give a brief introduction to these operations used in our models.

2.1.1 Convolution operation

The convolution operation is basically used to calculate the cross-correlation between the kernel weights and the image pixels (or previous layer output). It is an element-wise multiplication operation followed by a summation between the kernel feature map and the input, as shown in the equation 2.1.

$$y(N_i, C_{out_j}) = \sum_{k=0}^{C_{in}-1} \theta(C_{out_j}, k) * x(N_i, k) + b(C_{out_j}) \tag{2.1}$$

where, x is the input to the convolution layer and y is the output, θ are the kernel weights and b is the bias. The input size is (N, C_{in}, H, W) and output size is $(N, C_{out}, H_{out}, W_{out})$; N is the batch-size, C are the number of channels, and H, W are the input height and width.
The idea behind this operation is that the kernel weights are learned using back-propagation so that they extract relevant features from the input. A kernel's weights are spatially shared for all locations in the image. This is because a feature extracted by a kernel at a particular part in an image is relevant for other parts of the image as well. This is an important property of CNNs as it reduces the number of parameters. The convolution operation is performed as a matrix operation rather than an individual element-wise multiplication and summation in practice in almost all widely used deep learning libraries.

2.1.2 Batch Normalization

During the forward pass, the multiplication between weights and the input shifts the distribution along the direction of the weights. In a deep neural network, subsequent operations result in a huge shift in the distribution, which leads to loss in performance. This is called internal covariate shift. In order to tackle this problem, the batch normalization layer was proposed in [5]. Just like the input to the network must be normalized with zero mean and unit variance, the batchnorm layer normalizes the input to each layer by calculating the mean and variance in a batch-wise manner.

$$y = \frac{x - \mu(x)}{\sqrt{\sigma(x)}} * \gamma + \beta \tag{2.2}$$

where, μ and σ are the running estimates of mean and variance of the input x. And, γ and β are shifting and translation parameters learned during training. Using Batch Normalization in deep neural networks enables faster and more stable convergence. It also allows for higher learning rates and makes the overall learning process less dependent on weight initialization. Also, since batchnorm adds a little noise to the network, it provides some regularization effect.

2.1.3 ReLU Activation

The Rectified Linear Unit is an activation function most widely used in deep neural networks [6]. It is more closely related to the threshold function used in our brain to whether let a signal pass through a neuron or not. It can be regarded as a truncation operator applied element-wise on the input.

$$f(x) = \max(x, 0) \tag{2.3}$$

There is no parameter inside a ReLU layer, hence no need for parameter learning in this layer. It is used as a nonlinear activation function in neural networks, as shown in Figure 2.1 (left). In the case of images, the ReLU function will let through features that result positive for a certain patterns and will render other negative patterns to zero. In this way, the ReLU function induces sparsity in the model which reduces

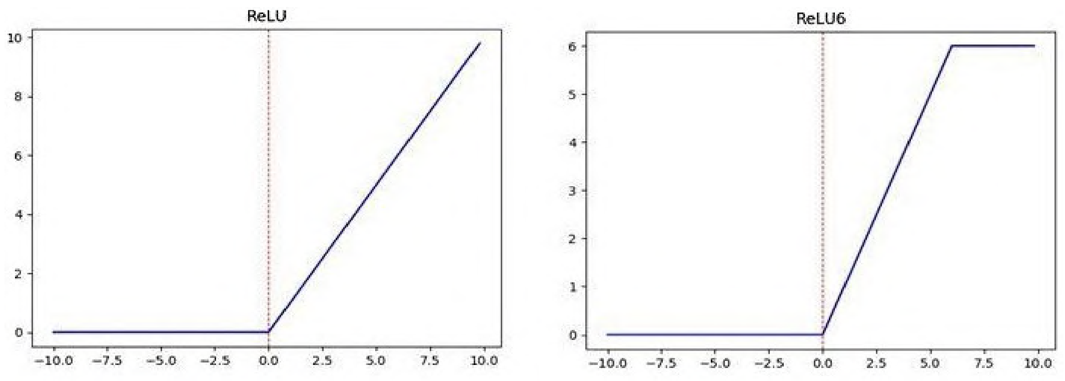


Figure 2.1: ReLU Activation Function

complexity and memory consumption.
Other popular activation functions such as sigmoid and tanh have a tendency of saturating at their extremas. Also, unlike these functions, ReLU does not suffer from vanishing gradient problem because its derivative is a step function that produces a constant 1 for non-zero activation values.

$$f'(x) = \begin{cases} 1, & \text{if} \quad x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

Although there are many different versions of ReLU, we use the ReLU6 version [7], which clips the activation at value equal to 6, as shown in Figure 2.1 (right). It is defined as:

$$f(x) = \min(\max(x, 0), 6) \tag{2.5}$$

The advantage of using ReLU6 is that since it clips values above 6 passing through to it, that encourages the model to learn sparse features earlier. The ReLU6 function can be generalized to ReLUn family of activation functions, where n can be any real number. Keeping $n = 6$ works best in practice [7].

2.2 Signal Processing

Signal processing is a subfield of electrical engineering that deals with feature extraction, modification, analysis and synthesis of data that can be represented in form of signals. Signal processing techniques have been widely used in almost every scientific and non-scientific field. Signal processing techniques can be categorized based on the types of signals to be processed.
In this thesis, we focus on signal processing techniques pertaining to audio and speech signals. The techniques used to process audio signals are mostly used for analysis and feature extraction. We employ some of these techniques to extract distinguishable features from audio signals. We process audio signals to extract features using the methods briefly discussed next.

2.2.1 Mel Frequency Cepstral Coefficients

The Mel-Frequency Cepstral Coefficients (MFCC) has been one of the standard signal/audio feature extraction technique [8] and has been successfully used to benchmark applications like speaker recognition [9], music information retrieval [10], speech recognition [11]. The development of MFCC was propelled by human auditory perception. MFCCs produce a compact representation of an audio signal. It differs from other cepstral features in the frequency bands which are on the mel-scale. The detailed five step procedure to extract MFCCs can be found in [12].

2.2.2 Gammatone Frequency Cepstral Coefficients

The Gammatone Frequency Cepstral Coefficients (GFCC) has also been a popular choice of feature extraction for audio/signal processing [13]. The gammatone filter is a linear filter that is outlined by an impulse response which is a product of a gamma distribution and sinusoidal tone. Hence, the name gammatone. It is especially advantageous to use GFCC with MFCC as they complement each other, due to the capability of GFCC being able to proficiently characterize transient sounds classes such as footsteps and gun-shots [14]. Detailed analysis of the benefits of combining MFCC and GFCC can be found in [15].

2.2.3 Constant Q-Transform

The Constant Q-transform is a time-frequency analysis technique that is particularly suitable for music audio signals [16–18]. It is essentially a Gabor wavelet transform, so unlike STFT, it has higher frequency resolution for lower frequencies and higher time resolution for higher frequencies. Due to this, it was shown in [19] that CQT outperformed standard MFCC feature extraction for ESC using CNNs. The results shown in [20], illustrated CQT's ability to capture low-to-mid level frequencies better than MFCC for audio scene classification, which is essentially the same task as ESC.

2.2.4 Chromagram

Another feature extraction technique that is popular with music information retrieval and processing is the Chromagram [21]. Chroma based features are especially useful for pitch analysis of audio signals. They can be used to distinguish among audio signals by assigning them pitch class profiles. This makes chromagrams particularly proficient in audio structure analysis [22]. We use the STFT (Short-time Fourier Transform) spectrogram to compute chroma features. The implementation has been derived from [23].

MFCC acts as the backbone by providing rich features, GFCC adds transient sound features, CQT contributes with better low-to-mid frequency range features and finally Chromagram provides pitch category analysis and signal structure information. Different feature extraction methods interpret audio information in different ways. The methods mentioned above represent audio information in contrasting ways with each of them giving distinct information. Some features, like the MFCC and GFCC, represent that amplitude spikes with high values, whereas CQT and Chromagram represent it with low values. The representation of MFCC is completely different as it provides some positive value in every region, with enough discrimination capabilities. On the other hand, the other features act as complementary features that eke out some additional distinguishable features.

2.3 Extreme Learning Machine

The Extreme Learning Machine is a supervised learning algorithm for single hidden layer feed forward neural networks proposed in [24]. In this method, the input weights are randomly initialized and the output weights are determined analytically. It is a non-gradient one-shot learning algorithm that does not require repeated iterations as in backpropagation. It is one of the fastest learning algorithms, however it is unstable.

2.3.1 The Theory of ELM

The input to the ELM, in this case, are the features extracted by the CNNs. Let it be represented as x_i, t_i, where x_i is the input feature instance and t_i is the corresponding class of the image. The inputs are connected to the hidden layer by randomly assigned weights w. The product of the inputs and their corresponding weights act as inputs to the hidden layer activation function. The hidden layer activation function is a nonlinear non-constant bounded continuous infinitely differentiable function that maps the input data to the feature space. There is a catalogue of activation functions from which we can choose according to the problem at hand. We ran experiments for all activation functions and the best performance was achieved with the multi-quadratics function:

$$f(x) = \sqrt{\|x_i - \mu_i\|^2 + a^2} \tag{2.6}$$

The hidden layer and the output layer are connected via weights β, which are to be analytically determined. The mapping from the feature space to the output space is linear. Now, with the inputs, hidden neurons, their activation functions, the weights connecting the inputs to the hidden layer and the output weights produce the final output function:

$$\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) = o_j \tag{2.7}$$

The output in Matrix form is:

$$H\beta = T \tag{2.8}$$

The error function used in Extreme Learning Machine is the Mean Squared error function, written as:

$$E = \sum_{j=1}^{N} (\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) - t_j)^2 \tag{2.9}$$

To minimize the error, we need to get the least-squares solution of the above linear system.

$$||H\beta^* - T|| = min_\beta ||H\beta - T|| \tag{2.10}$$

The minimum norm least-squares solution to the above linear system is given by:

$$\hat{\beta} = H^\dagger T \tag{2.11}$$

Properties of the above solution:

1. *Minimum Training Error:* The following equation provides the least-squares solution, which means the solution for $||H\beta - T||$, i.e. the error is minimum.
   $||H\beta^* - T|| = min_\beta ||H\beta - T||$

2. *Smallest Norm of Weights:* The minimum norm of least-squares solution is given by the Moore-Penrose pseudo inverse of H.
   $\hat{\beta} = H^\dagger T$

3. *Unique Solution:* The minimum norm least-squares solution of $H\beta = T$ is unique, which is:
   $\hat{\beta} = H^\dagger T$

Detailed mathematical proofs of these properties and the ELM algorithm can be found in [25].

2.4 Reinforcement Learning

Reinforcement Learning (RL) has been a subject of extensive research and applications in various real world domains such as Robotics, Games, Industrial Automation

and Control, System Optimization, Quality Control and Maintenance. Reinforcement Learning is a sub-field of Machine Learning which deals with learning to make appropriate decisions and take actions to achieve a goal. A Reinforcement Learning agent learns from direct interactions with an environment without requiring explicit supervision or a complete model of the environment. The agent interacts with the environment by performing actions. It receives feedback for it's actions in terms of reward (or penalty) from the environment and observes changes in the environment as a result of the actions it performs. These observations are called states of the environment and the agent interacts with the environment at discrete time intervals t by performing an action a_t in a state of the environment s_t, it transitions to a new state s_{t+1} (change in the environment) while receiving a reward r_t, with probability $P(s_{t+1}|s_t, a_t)$. The main aim of the agent is to maximize the cumulative reward over time through it's choice of actions.
A pictorial representation of the RL framework is shown in Figure 2.2. In the subsequent subsections, a brief presentation of the concepts and methods used in this thesis are explained.

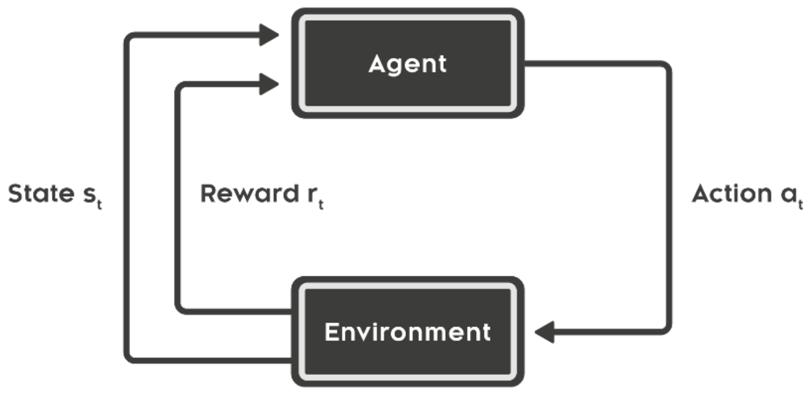


Figure 2.2: Reinforcement Learning Framework (the figure is taken from [1])

2.4.1 Markov Decision Process

The Reinforcement learning framework is formalised by Markov Decision Processes (MDP) which are used to define the interaction between a learning agent and its environment in terms of states, actions, and rewards [26]. An MDP consists of a tuple of $\langle S, A, P, R\rangle$ [1], where S is the state space, A is the action space, P is the transition probability from one state to the next, $P : S \times A \times S \longmapsto [0, 1]$ and R is the reward function, $R : S \times A \longmapsto \mathbb{R}$.
When state space S, action space A and rewards R consist of finite number of elements, s_{t+1} and r_{t+1} have well-defined discrete probability distributions which

depend only on the present state and action (Markov Property). This is represented as $p(s_{t+1}, r_{t+1}|s_t, a_t)$, where p determines the dynamics of the Markov Decision Process and where:

$$\sum_{s_{t+1}\in S}\sum_{r\in R} p(s_{t+1}, r_{t+1}|s_t, a_t) = 1, \forall s_t \in S, a_t \in A \tag{2.12}$$

p contains all the information about the MDP, so we can compute important aspects about the environment from p, like state transition probability and expected rewards for state-action pairs [1]:

$$P(s_{t+1}|s_t, a_t) = \sum_{r\in R} p(s_{t+1}, r_{t+1}|s_t, a_t) \tag{2.13}$$

$$r(s_t, a_t) = \mathbb{E}[r_t|s_t, a_t] = \sum_{r\in R} r \sum_{s_{t+1}\in S} p(s_{t+1}, r_{t+1}|s_t, a_t) \tag{2.14}$$

The equation 3, gives the immediate reward we expect to get when performing action a_t from state s_t. The agent tries to select actions that maximize the sum of rewards it expects to achieve, as time goes to infinity. But, in a dynamic and/or continuous Markov Decision Process, the notion of discounted rewards is used [1]:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.15}$$

where, γ is the discount factor and is in the range $[0, 1]$. If γ is near 0, then the agent puts emphasis on rewards received in the near future and if γ is near 1, then the agent also cares about rewards in the distant future.
In order to maximize G_t, the agent picks an action a_t when in a state s_t according to a policy function $\pi(s_t)$. A policy function is a probabilistic mapping from the state space to the action space, $S \to A$. The policy function outputs probabilities for taking each action in give state, so it can also be denoted as $\pi(a_t|s_t)$.

2.4.2 Q-Learning

Most of the Reinforcement Learning algorithms (value based) try to estimate the value function which gives an estimate of how good a state is for the agent to reside in. This is estimated according to the expected reward of a state under a policy and is denoted as $v_\pi(s)$:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|s_t] \tag{2.16}$$

Q-learning is a value based Reinforcement Learning algorithm that tries to maximize the q function [27]. The q function is a state-action value function and is denoted by $Q(s_t, a_t)$. It tries to maximize the expected reward give a state and action performed on that state:

$$Q(s_t, a_t) = \mathbb{E}[G_t | s_t, a_t] \tag{2.17}$$

According the Bellman Optimality equation [1], the optimal q function can be obtained by:

$$\begin{aligned} Q^*(s_t, a_t) &= \mathbb{E}[r_{t+1} + \gamma v^*(s_{t+1}) | s_t, a_t] \\ &= \sum_{s_{t+1}, r_t} p(s_{t+1}, r_t | s_t, a_t)[r_t + \gamma v^*(s)] \end{aligned} \tag{2.18}$$

where, $v^*(s_{t+1}) = \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$. And, a^* is the optimal action which results in maximum reward, the optimal policy is formed as $\arg\max_{a_t} \pi^*(a_t|s_t) = a^*$. This method was proposed in [27] which is tabular style Q-learning. The update rule for each time step of Q-learning is as follows:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \eta[r_t + \gamma \max_{a_t} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \tag{2.19}$$

Q-learning is an incremental dynamic programming algorithm that determines the optimal policy in a step-by-step manner. At each step t, the agent performs the following operations:

- Observes current state s_t.
- Selects and performs an action a_t.
- Observes the next state s_{t+1}.
- Receives the reward r_t.
- Updates the q-values $Q_t(s_t, a_t)$ using equation 2.19.

The q value function converges to the optimal value $Q_{t+1}(s_t, a_t) \to Q^*(s_t, a_t)$ as $t \to \infty$. Detailed convergence proof and analysis can be found in [27].
This tabular Q-learning method is used in our proposed approach to generate a Q-matrix for the shortest path to the exit based on the building model. In order to incorporate the shortest path information, this Q-matrix is used to pretrain the DQN models.

2.4.3 Deep Q Network

The tabular Q-learning approach works well for small environments, but becomes infeasible for complex environments with large multidimensional discrete or continuous state-action spaces. To deal with this problem, a parameterized version of the q function is used for approximation $Q(s_t, a_t; \theta) \approx Q^*(s_t, a_t)$. This way of function approximation was first proposed in [28].

Deep Neural Networks (DNNs) have become the predominant method for approximating complex intractable functions. They have become the defacto method for various applications such as image processing and classification [4, 29–34], speech recognition [35–41], and natural language processing [42–46]. DNNs have also been applied to reinforcement learning problems successfully by achieving noteworthy performance [47, 48].

The most noteworthy research in integrating deep neural networks and Q-learning in an end-to-end reinforcement learning fashion is the Deep Q-Networks (DQNs) [49, 50]. To deal with the curse of dimensionality, a neural network is used to approximate the parameterised Q-function $Q(s_t, a_t; \theta)$. The neural network takes a state as input and approximates Q-values for each action based on the input state. The parameters are updated and the Q-function is refined in every iteration through an appropriate optimizer like Stochastic Gradient Descent [51], RMSProp [52], Adagrad [53], Adam [54] etc. The neural network outputs q-values for each action for the input state and the action with the highest q-value is selected (There is another DQN architecture, which is less frequently used, that takes in the state and action as input and returns it's q-value as output).

The DQN can be trained by optimizing the following loss function:

$$L_i(\theta_i) = \mathbb{E}[(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) - Q(s_t, a_t; \theta_i))^2] \tag{2.20}$$

where, γ is the discount factor, θ_i and θ_{i-1} are the Q-network parameters at iteration i and $i-1$ respectively. In order to train the Q-network, we require a target to calculate loss and optimize parameters. The target q-values are obtained by holding the parameters θ_{i-1} fixed from the previous iteration.

$$y = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) \tag{2.21}$$

where, y is the target for the next iteration to refine the Q-network. Unlike supervised learning where the optimal target values are known and fixed prior to learning, in DQN the approximate target values y, which depend on network parameters, are used to train the Q-network. The loss function can be rewritten as:

$$L_i(\theta_i) = \mathbb{E}[(y - Q(s_t, a_t; \theta_i))^2] \tag{2.22}$$

The process of optimizing the loss function $L_i(\theta_i)$ at the i^{th} iteration by holding the parameters from the previous iteration θ_{i-1} fixed, to get target values, results in a sequence of well-defined optimization time-steps. By differentiating the loss function in equation 2.22, we get the following gradient:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}[(y - Q(s_t, a_t; \theta_i))\nabla_{\theta_i} Q(s_t, a_t; \theta_i)] \tag{2.23}$$

Instead of computing the full expectation of the above gradient, we optimize the loss function using an appropriate optimizer. The DQN is a model-free algorithm since it directly solves tasks without explicitly estimating the environment dynamics. Also, DQN is an off-policy method as it learns a greedy policy $a = \arg\max_{a_{t+1}} Q(s, a_{t+1}; \theta)$, while following an ϵ-greedy policy for sufficient exploration of the state space. One of the drawbacks of using a nonlinear function approximator like neural network is that it tends to diverge and is quite unstable for reinforcement learning. The problem of instability arises mostly due to: correlations between subsequent observations and that small changes in q-values can significantly change the policy and the correlations between q-values and target values.

The most well-known and simple technique to alleviate the problem of instability is the experience replay [55]. At each time-step, a tuple consisting of the agent's experience $E_t = (s_t, a_t, r_t, s_{t+1})$ is stored in a replay memory over many episodes. A minibatch of these tuples is randomly drawn from the replay memory to update the DQN parameters. This ensures that the network isn't trained on a sequence of observations (avoiding strong correlations between samples and reducing variance between updates) and it increases sample efficiency. This technique greatly increases stability of DQN.

2.4.4 Double DQN

Q-learning and DQN are capable of achieving performance beyond the human level on many occasions. However, in some cases Q-learning performs poorly and so does its deep neural network counterpart DQN. The main reason behind such poor performance is that Q-learning tends to overestimate action values. These overestimations are caused due to a positive bias that results from the $\max$ function in

Q-learning and DQN updates which outputs the maximum action value as an approximation of the maximum expected action value.
The Double Q-learning method was proposed in [56] to alleviate this problem and later extended to DQN [57] to produce the Double DQN (DDQN) method. Since Q-learning uses the same estimator to select and evaluate an action, which results in overoptimistic action values, we can interpret it as a single estimator. In Double Q-learning, the task of evaluation and selection is decoupled by using double estimator approach consisting of two functions: Q^A and Q^B. The Q^A function is updated with a value from the Q^B function for the next state and the Q^B function is updated with a value from the Q^A function for the next state.
Let,

$$a^* = \arg\max_{a_t} Q_t^A(s_{t+1}, a_t) \tag{2.24}$$

$$b^* = \arg\max_{a_t} Q_t^B(s_{t+1}, a_t) \tag{2.25}$$

Then,

$$Q_{t+1}^A(s_t, a_t) = Q_t^A(s_t, a_t) + \eta[r_t + \gamma Q_t^B(s_{t+1}, a^*) - Q_t^A(s_t, a_t)] \tag{2.26}$$

$$Q_{t+1}^B(s_t, a_t) = Q_t^B(s_t, a_t) + \eta[r_t + \gamma Q_t^A(s_{t+1}, b^*) - Q_t^B(s_t, a_t)] \tag{2.27}$$

where, a^* is the action with the maximum q-value in state s_{t+1} according to the Q^A function and b^* is the action with the maximum q-value in state s_{t+1} according to the Q^B function.
The double estimator technique is unbiased which results in no overestimation of action values, since action evaluation and action selection is decoupled into two functions that use separate $\max$ function estimates of action values. In fact, thorough analysis of Double Q-learning in [56] shows that it sometimes might underestimate action values.
The Double Q-learning algorithm was adapted for large state-action spaces in [57] by forming the Double DQN method in a similar way as DQN. The two Q-functions (Q^A and Q^B) can be parameterised by two sets of weights θ and θ'. At each step, one set of weights θ is used to update the greedy policy and the other θ' to calculate it's value. For Double DQN, equation 2.21 can be written as:

$$y = r_t + \gamma Q(s_{t+1}, \arg\max_{a} Q(s_{t+1}, a_t; \theta_i); \theta_i') \tag{2.28}$$

The first set of weights θ are used to determine the greedy policy just like in DQN. But, in Double DQN, the second set of weights θ' is used for an unbiased value

estimation of the policy. Both sets of weights can be updated symmetrically by switching between θ and θ'.
The target value network in DQN can be used as the second Q-function instead of introducing an additional network. So, the weights at the i^{th} iteration are used to evaluate the greedy policy and the weights at the previous iteration to estimate it's value. The update rule remains the same as DQN, while changing the target as:

$$y = r_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a_t; \theta_i); \theta_{i-1}) \tag{2.29}$$

Note that in both DQN and DDQN, the target network uses the parameters of the previous iteration $i-1$. However, to generalise, the target network can use parameters from the any previous $(i-k)^{th}$ iteration. Then, the target network parameters are updated periodically with the copies of the parameters of the online network.

2.4.5 Dueling DQN

In quite a few RL applications, it is sometimes unnecessary to estimate the value of each action. In many states, the choice of action has no consequence on the outcome. A new architecture for model-free Reinforcement Learning, called the dueling architecture, is proposed in [58]. The dueling architecture explicitly separates state values and action advantage values into two streams which share a common feature extraction backbone neural network. The architecture is similar to that of the DQN and DDQN architectures; the difference being that instead of a single stream of fully connected layers, there are two streams providing estimates of the value and state-dependent advantage functions. The two streams are combined at the end producing a single Q-function.
One stream outputs a scalar state value, while the other outputs an advantage vector having dimensionality equal to number of actions. Both the streams are combined at the end to produce the Q-function estimate. The combining module at the end can simply aggregate the value and advantage estimates as:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \mathcal{A}(s_t, a_t; \theta, \alpha) \tag{2.30}$$

where, θ are the parameters of the lower layers of the neural network (before streams are split); α and β are the parameters of the advantage and value function streams. However, such an aggregation of streams would require $V(s_t; \theta, \beta)$ to be replicated as many times as the dimensionality of $\mathcal{A}(s_t, a_t; \theta, \alpha)$. Also, value and advantage estimates cannot be uniquely recovered given the estimated Q-function.

One way of addressing these issues, proposed in [58], is to force the advantage function estimator to have zero value at the selected action. This aggregation is implemented in the combining module as:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \left(\mathcal{A}(s_t, a_t; \theta, \alpha) - \max_{a_{t+1} \in A} \mathcal{A}(s_t, a_{t+1}; \theta, \alpha)\right) \quad (2.31)$$

Now, for a chosen action (action with max Q-function), $a^* = \arg\max_{a_{t+1} \in A} Q(s_t, a_{t+1}; \theta, \alpha, \beta)$, putting in equation 2.31, we get $Q(s_t, a^*; \theta, \alpha, \beta) = V(s_t; \theta, \beta)$. Hence, the two streams can be uniquely recovered. In [58], another way of aggregation is proposed which eliminates the $\max$ operator.

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \left(\mathcal{A}(s_t, a_t; \theta, \alpha) - \frac{1}{|A|} \sum_{a_{t+1}} \mathcal{A}(s_t, a_{t+1}; \theta, \alpha)\right) \quad (2.32)$$

where, $|A|$ is the number of actions. Even though value and advantage estimates are now off-target by a constant, this way of aggregation improves stability by capping the changes in the advantage estimates by their mean and enhances overall performance.

2.5 Summary

In this chapter, we have discussed the several background concepts underlining this thesis. The thesis consists of core deep learning, reinforcement learning and neural network concepts, which have been thoroughly explained in this chapter. The preliminaries described in this chapter have been used throughout the thesis. These basic and complex concepts of AI act as a base and the contributions of this thesis are built upon them. We have described Convolutional Neural Networks (Paper A, B, D, E, F), Reinforcement Learning, especially Q-learning and variants (Paper C), the Extreme Learning Machine algorithm (Paper B) and some signal processing methods (Paper E, F).

Chapter 3

Contributions

In this chapter, we explain and elaborate on the contributions of this thesis. As mentioned earlier, we divide the fire emergency management procedure into three phases: Detection, Analysis and Evacuation. The main contribution of this thesis is to find AI based solutions for fire emergencies that can replace current methods by eliminating their shortcomings and outperforming them. Also, we aim to reduce risk for fire-fighters by providing relevant and necessary information without any human intervention.
We use image classification for visual detection and sound classification for audio detection of a fire emergency. In the analysis phase, we use semantic segmentation to visually analyse the emergency situation by identifying inflammable objects and people in the environment. And finally, in the evacuation phase, we employ reinforcement learning to learn shortest paths to the nearest exit from each room to evacuate all people from the building. We design our own reinforcement learning simulator that consists of realistic fire scenarios. We elaborate our contributions for each of these phases in the following sections.

3.1 Detection

In the detection phase, the main task is to detect whether there is an emergency situation or is everything normal. We propose two ways of doing this. One is using visual information like images to classify whether there is a fire emergency or not. Another way is to use audio information to detect an emergency situation. Using the audio information way, we build a general purpose emergency detection system since we can classify all sounds that depict an emergency into a single class. On the other hand, we specialize the visual detection system to only detect fire emergencies.

Both ways of detection have their own advantages and disadvantages. Some information is most readily available visually (e.g., the extent of a fire), while other information is perhaps best captured through sound (e.g., screaming). In some cases, changes in audio signal might occur first. Whereas, in other cases, a change in the visual representation of the environment might take place before any other changes. We have therefore investigated methods for using both kinds of information in this thesis.

3.1.1 Visual Detection

Detecting fire in images using image processing and computer vision techniques has gained a lot of attention from researchers during the past few years. Indeed, with sufficient accuracy, such systems may outperform traditional fire detection equipment. One of the most promising techniques used in this area is Convolutional Neural Networks (CNNs). However, the previous research on fire detection with CNNs has only been evaluated on balanced datasets, which may give misleading information on real-world performance, where fire is a rare event.

There have been many innovative techniques proposed in the past to build an accurate fire detection system which are broadly based on image processing and computer vision techniques. The state-of-the-art vision-based techniques for fire and smoke detection have been comprehensively evaluated and compared in [59]. The colour analysis technique has been widely used in the literature to detect and analyse fire in images and videos [60–63]. On top of colour analysis, many novel methods have been used to extract high level features from fire images like texture analysis [61], dynamic temporal analysis with pixel-level filtering and spatial analysis with envelope decomposition and object labelling [64], fire flicker and irregular fire shape detection with wavelet transform [62], etc. These techniques give adequate performance but are outperformed by Machine Learning techniques. A comparative analysis between colour-based models for extraction of rules and a Machine Learning algorithm is done for the fire detection problem in [65]. The machine learning technique used in [65] is Logistic Regression which is one of the simplest techniques in Machine Learning and still outperforms the colour-based algorithms in almost all scenarios. These scenarios consist of images containing different fire pixel colours of different intensities, with and without smoke.

Instead of using many different algorithms on top of each other to extract relevant features, we can use a network that learns relevant features on its own. Neural networks have been successfully used in many different areas such as Natural Language Processing, Speech Recognition, Text Analysis and especially Image Classification. Extracting relevant features from images is the key to accurate classification and analysis which is why the problem of fire detection is ideally suited for Deep Learning. Deep Neural Networks are used to automatically 'learn' hierarchy of pertinent features from data without human intervention and the type of neural network ideally suited for image classification is the Convolutional Neural Networks (CNN).

Therefore, our approach is to employ state-of-the-art CNNs to distinguish between images that containing fire and images that do not and build an accurate fire detection system. To make these models more robust, we use a custom-made image dataset containing images with numerous scenarios.

We improve our fire detection model further by using hybrid models. We propose to use hybrid CNN-ELM and CNN-SVM models to outperform Deep CNNs. Such hybrid models have been used in the past for image classification, but the novelty of our approach lies in using state-of-the-art Deep CNNs like VGG16 and Resnet50 as feature extractors and then remove some/all fully connected layers with an ELM classifier. This models outperform Deep CNNs in terms of accuracy, training time and size of the network. We also compare the CNN-ELM model with another hybrid model, CNN-SVM and show that the CNN-ELM model gives the best performance.

3.1.1.1 Related Work: CNNs for Fire Detection

There have been many significant contributions from various researchers in developing a system that can accurately detect fire in the surrounding environment. But, the most notable research in this field involves Deep Convolutional Neural Networks (DCNN). DCNN models are currently among the most successful image classification models which makes them ideal for a task such as Fire detection in images. This has been demonstrated by previous research published in this area.

In [66], the authors use CNN for detection of fire and smoke in videos. A simple sequential CNN architecture, similar to LeNet-5 [32], is used for classification. The authors quote a testing accuracy of 97.9% with a satisfactory false positive rate.

Whereas in [67], a very innovative cascaded CNN technique is used to detect fire in an image, followed by fine-grained localisation of patches in the image that contain the fire pixels. The cascaded CNN consists of AlexNet CNN architecture [34] with pre-trained ImageNet weights [29] and another small network after the final pooling layer which extracts patch features and labels the patches which contain fire. Different patch classifiers are compared.

The AlexNet architecture is also used in [68] which is used to detect smoke in images. It is trained on a fairly large dataset containing smoke and non-smoke images

for a considerably long time. The quoted accuracies for large and small datasets are 96.88% and 99.4% respectively with relatively low false positive rates.
Another paper that uses the AlexNet architecture is [69]. This paper builds its own fire image and video dataset by simulating fire in images and videos using Blender. It adds fire to frames by adding fire properties like shadow, fore-ground fire, mask etc. separately. The animated fire and video frames are composited using OpenCV [70]. The model is tested on real world images. The results show reasonable accuracy with high false positive rate.
As opposed to CNNs which extract features directly from raw images, in some methods image/video features are extracted using image processing techniques and then given as input to a neural network. Such an approach has been used in [71]. The fire regions from video frames are obtained by threshold values in the HSV colour space. The general characteristics of fire are computed using these values from five continuous frames and their mean and standard deviation is given as input to a neural network which is trained using back propagation to identify forest fire regions. This method performs segmentation of images very accurately and the results show high accuracy and low false positive rates.
In [72], a neural network is used to extract fire features based on the HSI colour model which gives the fire area in the image as output. The next step is fire area segmentation where the fire areas are roughly segmented and spurious fire areas like fire shadows and fire-like objects are removed by image difference. After this the change in shape of fire is estimated by taking contour image difference and white pixel ratio to estimate the burning degree of fire, i.e. no-fire, small, medium and large. The experimental results show that the method is able to detect different fire scenarios with relatively good accuracy.
All the research work done in this area has been exemplary. But, there are some issues associated with each of them that we try to alleviate in our work. We use a dataset that consists of images that we have handpicked from the internet. The dataset contains images that are extremely hard to classify which results in poor generalization. The dataset also contains many different scenarios and is highly unbalanced to replicate real world behaviour. We propose to use state-of-the-art pre-trained DCNN models. The reason behind using such complex models is explained in the next section. We also modify these models to improve accuracy at the cost of training time.

3.1.1.2 Related Work: Hybrid models for Image classification

The classifier part in a Deep CNN is a simple fully connected perceptron with a softmax layer at the end to output probabilities for each class. This section of the CNN has a high scope for improvement. Since it consists of three to four fully connected layers containing thousands of neurons, it becomes harder and slower to train it. Even with pre-trained models that require fine tuning of these layers. This has led to the development of hybrid CNN models, which consist of a specialist classifier at the end.

Some of the researchers have employed the Support Vector Machine (SVM) as the final stage classifier [73–77]. In [73], the CNN-SVM hybrid model is applied to many different problems like object classification, scene classification, bird subcategorization, flower recognition etc. A linear SVM is fed 'off the shelf convolutional features' from the last layer of the CNN. This paper uses the OverFeat network [78] which is a state-of-the-art object classification model. The paper shows, with exhaustive experimentation, that extraction of convolutional features by a deep CNN is the best way to obtain relevant characteristics that distinguishes an entity from another.

The CNN-SVM model is used in [74] and successfully applied to visual learning and recognition for multi-robot systems and problems like human-swarm interaction and gesture recognition. This hybrid model has also been applied to gender recognition in [75]. The CNN used here is the AlexNet [34] pre-trained with ImageNet weights. The features extracted from the entire AlexNet are fed to an SVM classifier. A similar kind of research is done in [76], where the softmax layer and the cross-entropy loss are replaced by a linear SVM and margin loss. This model is tested on some of the most well known benchmark datasets like CIFAR-10, MNIST and Facial Expression Recognition challenge. The results show that this model outperforms the conventional Deep CNNs.

In 2006, G.B. Huang introduced a new learning algorithm for a single hidden layer feedforward neural network called the Extreme Learning Machine [24, 25]. This technique was many times faster than backpropagation and SVM, and outperformed them on various tasks. The ELM randomly initializes the input weights and analytically determines the output weights. It produces a minimum norm least squares solution which always achieves lowest training accuracy, if there are enough number of hidden neurons. There have been many variants of ELM depending upon a specific application, which have been summarised in [79].

This led to the advent of CNN-ELM hybrid models, which were able to outperform the CNN-SVM models on various applications. The major advantage of CNN-

ELM models is the speed of convergence. In [80], the CNN-ELM model is used for Wireless Capsule Endoscopy (WCE) image classification. The softmax classifier of a CNN is replaced by an ELM classifier and trained on the feature extracted by the CNN feature extractor. This model is able to outperform CNN-based classifiers.
The CNN-ELM model has also been used for handwritten digit classification [81, 82]. In [81], a 'shallow' CNN is used for feature extraction and ELM for classification. The shallow CNN together with ELM speeds up the training process. Also, various weight initialization strategies have been tested for ELM with different receptive fields. Finally, two strategies, namely the Constrained ELM (C-ELM) [83] and Computed Input Weights ELM (CIW-ELM) [84] are combined in a two layer ELM structure with receptive fields. This model was tested on the MNIST dataset and achieved 0.83% testing error. In [82], a deep CNN is used for the same application and tested on the USPS dataset.
A shallow CNN with ELM is tested on some benchmark datasets like MNIST, NORB-small, CIFAR-10 and SVHN with various hyper parameter configurations in [85]. Another similar hybrid model that uses CNN features and Kernel ELM as classifier is used in [86] for age estimation using facial features. Another application where a CNN-ELM hybrid model has been applied is the traffic sign recognition [87].
A different strategy of combining CNN feature extraction and ELM learning is proposed in [88]. Here, an ELM with single hidden layer is inserted after every convolution and pooling layer and at the end as classifier. The ELM is trained by borrowing values from the next convolutional layer and each ELM is updated after every iteration using backpropagation. This interesting architecture is applied to the application of lane detection and achieves excellent performance.
A comparative analysis of the CNN-ELM and CNN-SVM hybrid models for object recognition from ImageNet has been illustrated in [89]. Both these models were tested for object recognition from different sources like Amazon, Webcam, Caltech and DSLR. The final results show that the CNN-ELM model outperforms the CNN-SVM model on all datasets and using Kernel ELM further increases accuracy.
Using ELM as a final stage classifier does not end at image classification with CNNs. They have also been used with DBNs for various applications [90, 91].

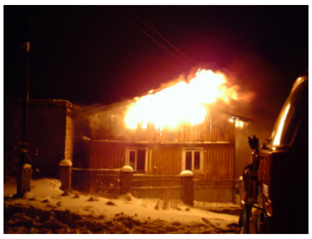
3.1.1.3 Fire Images Dataset

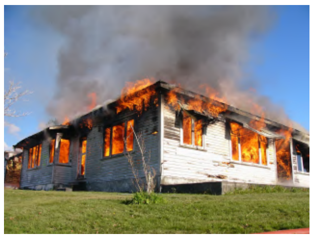
Since there is no benchmark dataset for fire detection in images, we created our own dataset by handpicking images from the internet. [1]This dataset consists of 651 images which is quite small in size but it enables us to test the generalization capabilities and the effectiveness and efficiency of models to extract relevant features from images when training data is scarce. The dataset is divided into training and testing sets. The training set consists of 549 images: 59 fire images and 490 non-fire images. The imbalance is deliberate to replicate real world situations, as the probability of occurrence of fire hazards is quite small. The datasets used in previous papers have been balanced which does not imitate the real world environment. The testing set contains 102 images: 51 images each of fire and non-fire classes. As the training set is highly unbalanced and the testing set is exactly balanced, it makes a good test to see whether the models are able to generalize well or not. For a model with good accuracy, it must be able to extract the distinguishing features from the small amount of fire images. To extract such features from small amount of data the model must be deep enough. A poor model would just label all images as non-fire, which is the case shown in the results.
Apart from being unbalanced, there are a few images that are very hard to classify. The dataset contains images from all scenarios like fire in a house, room, office, forest fire, with different illumination intensity and different shades of red, yellow and orange, small and big fires, fire at night, fire in the morning; non-fire images contain a few images that are hard to distinguish from fire images like a bright red room with high illumination, sunset, red coloured houses and vehicles, bright lights with different shades of yellow and red etc.
The Figures 3.1 show the fire images with different environments: indoor, outdoor, daytime, night-time, forest fire, big and small fire. And the Figures 3.2 show the non-fire images that are difficult to classify. Considering these characteristics of our dataset, detecting fire can be a difficult task. We have made the dataset available online so that it can be used for future research in this area.

3.1.1.4 Deep Convolutional Neural Networks for Fire Detection

We propose to use two Deep CNN architectures that have outperformed the AlexNet on the ImageNet dataset, namely VGG16 [30] and Resnet50 [31]. We use these models with pre-trained ImageNet weights. This helps greatly when there is lack of training data. So, we just have to fine-tune the fully-connected layers on our dataset.

[1]The dataset is available here: `https://github.com/UIA-CAIR/Fire-Detection-Image-Dataset`


(a)


(b)

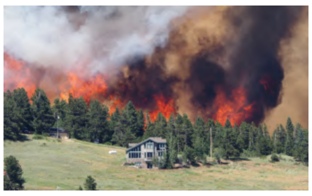
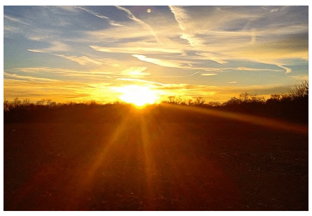
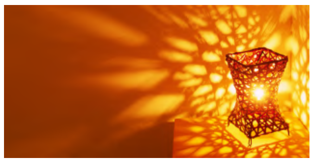

(c)

Figure 3.1: Examples of Fire Images


(a)


(b)

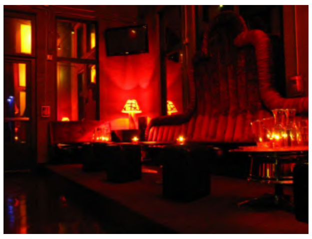
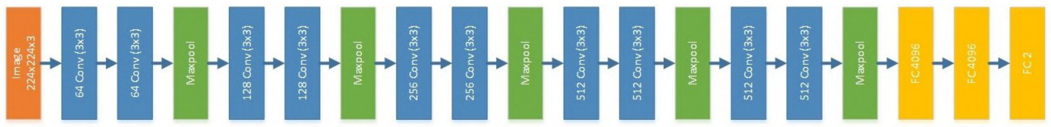

(c)

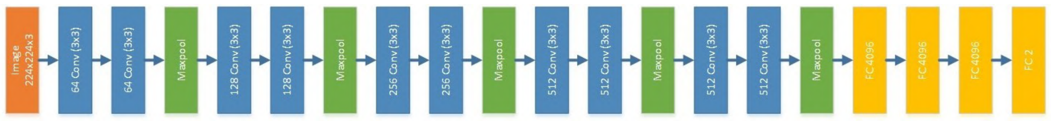
Figure 3.2: Examples of Non-Fire Images that are difficult to classify

VGG16: The VGG16 architecture was proposed by the Visual Geometry Group at the University of Oxford [30]. The main purpose of the paper was to investigate the effect of depth in CNN models. They developed a number of models with different depths ranging from 11 layers to 19 layers and tested them on different tasks. The results on these tasks show that increasing depth also increases performance and accuracy. The 19 layer architecture, VGG19 won the ImageNet challenge in 2014, but the 16 layer architecture, VGG16 achieved an accuracy which was very close to VGG19. Both the models are simple and sequential. The 3x3 convolution filters are used in the VGG models which is the smallest size and thus captures local features. The 1x1 convolutions can be viewed as linear transformations and can also be used for dimensionality reduction. We choose the VGG16 over the VGG19 because it takes less time to train and the classification task in hand is not as complex as ImageNet challenge. Both the models have the same number of fully connected layers, i.e. 3, but differ in the number of 3x3 filters.

VGG16 (modified): In this work, we also test a modified version of VGG16 which consists of 4 fully connected layers, fine-tuned on the training data, which was able to increase the accuracy of classification. We also tested with more fully connected layers but the increase in accuracy was overshadowed by the increase in training time. The Figures 3.3 a and 3.3 b show the original and modified



(a) VGG16 Architecture



(b) Modified VGG16 Architecture

Figure 3.3

VGG16 architectures respectively.

ResNet50: After the success of the VGG architectures, it was established that deeper models outperform shallower networks. But, the problem with making models deeper was the difficulty in training them because model complexity increases as the number of layers increase. This issue was addressed by Microsoft Research, who proposed extremely deep architectures but with lower complexity [31]. They introduced a new framework of learning to ease training of such deep networks. This is called Residual learning and hence the models that employed this framework are called Residual Networks. Residual Learning involves learning residual functions. If a few stacked layers can approximate a complex function, $F(x)$ where, x is the input to the first layer, then they can also approximate the residual function $F(x) - x$. So, instead the stacked layers approximate the residual function$G(x) = F(x) - x$, where the original function becomes $G(x) + x$. Even though both can capable of approximating the desired function, the ease of training with residual functions is better. These residual functions are forwarded across layers in the network using identity mapping shortcut connections. The ImageNet 2015 results show that Resnet achieves the lowest error rates in image classification. The Resnet architectures consist of networks of various depths: 18-layers, 34-layers, 50-layers, 101-layers and 152-layers. We choose the architecture with intermediate depth, i.e. 50 layers. The Resnet consists of 3x3 and 1x1 filters, pooling layers and residual connections and a single softmax layer at the end.

ResNet50 (modified): We also test a modified Resnet model by adding a fully connected layer fine-tuned on the training data, which increase accuracy further. We did not add any more fully connected layers since the model is already quite deep and takes a long time to train. The Figures 3.4 a and 3.4 b show the original and

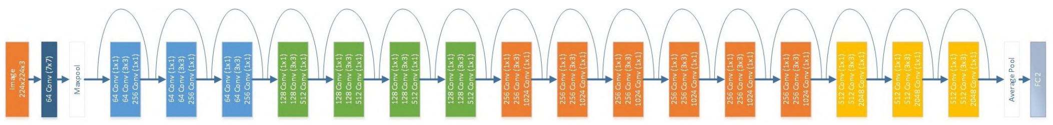


Image 224x224x3
64 Conv (7x7)
Maxpool
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
Average Pool
FC 2

(a) Resnet50 Architecture

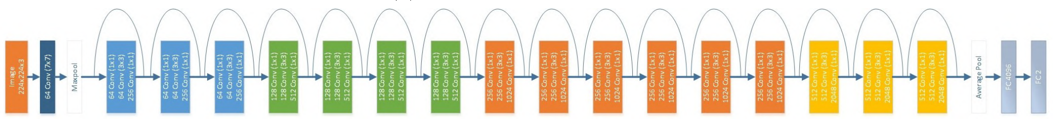


Image 224x224x3
64 Conv (7x7)
Maxpool
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
Average Pool
FC 4096
FC 2

(b) Modified Resnet50 Architecture

Figure 3.4

modified Resnet50 architectures respectively.

3.1.1.5 The Hybrid Model: Deep CNN-ELM for Fire Detection

We propose to use a hybrid architecture for fire detection in images. Instead of using a simple CNN as feature extractor, we employ state-of-the-art Deep CNNs like the VGG16 and Resnet50. Usually, only the softmax classifier is replaced by another classifier (ELM or SVM) in a CNN to create a hybrid model. But, we go one step further by replacing the entire fully connected multi-layer perceptron with a single hidden layer ELM. This decreases the complexity of the model even further.
Both the VGG16 and Resnet50 extract rich features from the images. These features are fed to the ELM classifier which finds the minimum norm least squares solution. With enough number of hidden neurons, the ELM outperforms the original VGG16 and Resnet50 networks. Both VGG16 and Resnet50 are pre-trained with ImageNet weights. So, only the ELM classifier is trained on the features extracted by the CNNs.
Apart from fast training and accurate classification, there is another advantage of this model. This hybrid model does not require large training data. In fact, our dataset consists of just 651 images, out of which the ELM is trained on 60% of images only. This shows its robustness towards lack of training data. A normal Deep CNN would require much higher amount of training data to fine-tune its fully-connected layers and the softmax classifier. Even the pre-trained VGG16 and Resnet50 models required at least 80% training data to fine-tune their fully-connected layers.

3.1.2 Audio Detection

Apart from detection using visual information, audio information can also be used to detect potential emergency situations. A more general purpose emergency detection approach is proposed using Environment Sound Classification (ESC). While building a model for emergency detection using audio signals, we were able to create a novel CNN model using multiple signal feature channels and an attention mechanism that set the new state-of-the-art on the most widely used benchmark ESC datasets. So, we first explain our novel ESC model and then show its application to emergency detection.

ESC is one of the most important tasks that deals with distinguishing between sounds from the real environment. It is a complex task that involves classifying a sound event into an appropriate class such as siren, dog barking, airplane, people talking etc.

The most successful ESC models consist of one or more standard audio feature extraction techniques and deep neural networks. We explore the idea of employing multiple feature extraction techniques like the Mel-frequency Cepstral Coefficients (MFCC) [8], Gammatone Frequency Cepstral Coefficients (GFCC) [13], Constant Q-Transform (CQT) [16], Chromagram [21] and stack them to create a multiple channel input to our classifier.

After feature extraction, the next stage is classification. Many machine learning algorithms have been used to classify sound, music or audio events. However, in the ESC task, Deep CNNs have been able to outperform other techniques, as evident from the previous ESC models. We also employ a Deep CNN for classification. However, we split between time and frequency domain feature processing by using separable convolutions [92] with different kernel sizes. Also, we use max pooling across only one of the domains at a time, until after the last set of convolutional layers to combine time and frequency domain features. This enables processing time and frequency domain features separately and then combining them at a later stage.

Along with the model, we also design a novel attention module that enables both spatial and channel attention. In order to achieve both spatial and channel attention with the same module, we need an attention weight matrix with dimensions equal to the DCNN block output. So that, each output feature map in each channel has it's own attention weights. We use the depthwise separable convolution [93] to achieve attention with minimal increase in number of parameters.

For the audio detection phase, we use a smaller version of the model proposed above. We propose a general purpose emergency detector based on our ESC model that classifies environmental sounds into emergency and non-emergency categories. The system is emergency type agnostic, so it can be used to detect any kind of crisis using audio signals from the environment only. To the best of our knowledge, this is the first general purpose emergency sound detection system.

3.1.2.1 Related Work: Environment Sound Classification

There have been several innovative and high performance approaches proposed for the task of environmental sound classification (ESC). In [94], a deep CNN was shown to give competitive results for the ESC tasks by thorough and exhaustive experimentation on the three benchmark datasets.
In [95], phase encoded filterbank energies (PEFBEs) was proposed as a novel feature extraction technique. Finally, a score-level fusion of FBEs and PEFBEs with a CNN classifier achieved best performance.
In the second version of the EnvNet, called EnvNetv2 [96], the authors employed a mechanism called Between Class (BC) learning. In BC learning, two audio signals from different classes are mixed with each other with a random ratio. The CNN model is then fed the mixed sound as input and trained to output this mixing ratio.
An unsupervised approach of learning a filterbank from raw audio signals was proposed in [97]. Convolutional Restricted Boltzmann Machine (ConvRBM), which is an unsupervised generative model, was trained to raw audio waveforms. A CNN is used as a classifier along with ConvRBM filterbank and score-level fusion with Mel filterbank energies. Their model achieves 86.5% on the ESC-50 dataset.
A novel data augmentation technique, called mixup, was proposed in [98]. It consists of mixing two audio signals and their labels, in a linear interpolation manner, where the mixing is controlled by a factor λ. In this way, their model achieves 83.7% accuracy on the UrbanSound8K dataset. We employ the mix-up data augmentation in our work to boost our model's performance.
A complex two stream structure deep CNN model was proposed in [99]. It consists of two CNN streams. One is the LMCNet which works on the log-mel spectrogram, chroma, spectral contrast and tonnetz features of audio signals and the other is the MCNet which takes MFCC, chroma, spectral contrast and tonnetz features as inputs. The decisions of the two CNNs are fused to get the final TSDCNN-DS model. It achieves 97.2% accuracy on the UrbanSound8K dataset.
There have also been a few contributions towards the ESC task that consist of attention based systems. In [100], a combination of two attention mechanisms, channel and temporal, was proposed. The temporal attention consists of 1×1 convolution for feature aggregation followed by a small CNN to produce temporal attention weights. On the other hand, channel attention consists of a bank of fully connected layers to produce the channel attention map. Using two separate attention models makes the system very complex and increases the number of parameters by a lot. We perform spatial and channel attention with just one depthwise convolutional layer.

A multi-stream network with temporal attention for the ESC task was proposed in [101]. The model consists of three streams with each stream receiving one of the three stacked inputs: raw waveform, STFT (Short-time Fourier Transform) and delta STFT. A temporal attention model received the inputs directly and propagated it's output to the main models intermediate layers. Here, again, the model is too complex and also, the attention block doesn't receive any intermediate feedback from the main model.

3.1.2.2 Related Work: Emergency Audio Classification

Different types of emergencies require different kinds of sensory detection systems. However, some detection systems are applicable to several types of emergencies. For example, vision based systems can be used to detect several kinds of emergencies, especially fire [102, 103], flood [104], earthquakes [105], droughts [106], avalanches [107] etc. A thorough analysis of vision based techniques to detect emergency situations can be found in [108].

There are also a few specialist detectors that detect specific types of emergencies that cannot be detected by conventional means. In [109], volatile chemical compounds are identified using a method based on portable gas chromatograph-mass spectrometer. Another specialist detector was proposed in [110] for human activity recognition. It used Bayesian networks and rules based stochastic context-free grammars to detect abnormal behaviours among monitored humans. Another technique proposed in [111], that monitors human activities was used for fall detection. It uses sensory data like acceleration and Euler angle with monitored parameters such as heart-rate and ADLs (activity of daily lives).

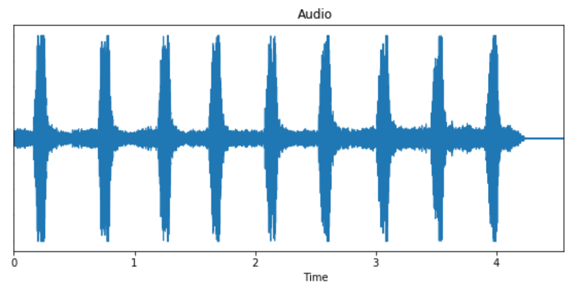
A Convolutional Neural Network (CNN) was used in [112] to detect emergency weather situations by processing data from climate datasets. The method demonstrated the ability of deep neural networks to learn complex weather patterns to predict extreme weather conditions. Along with Bayesian hyperparameter optimization, the weather model was able to achieve high performance. A very important work regarding the analysis of reliability of multi-state emergency detection systems was performed in [113]. Wireless sensor network based approaches to detect fire emergencies have also been popular in recent years [114–116]. The general idea uses a distributed system of nodes, where each node is capable of classifying fire emergencies. Once the fire is detected, the cluster-head is alerted and routed to other cluster-heads via gateways to alert the firefighters. Each node consists of a fire detection algorithm and sometimes a neural network that has been trained on historical fire data.

A research work similar to our proposed task was proposed in [117]. They use a perception sensor network that detects screaming persons. It also uses audio-visual information for sound source localization. However, unlike our work, it does not generalize to all major emergency situations and does not employ deep learning. Also, our method uses auditory information only. Also, in [118], a core signal processing approach of using mechanical resonant filters is used to detect sirens. A siren detection for system ambulance using fast fourier transform was proposed in [119]. Another emergency siren and horn detection system was proposed in [120]. They propose to treat the audio spectrograms as an image and apply semantic segmentation using the U-net [121] to detect and de-noise sirens in traffic. They use another CNN to localize the siren sounds, which makes this system very computationally complex.
Unlike these methods, our proposed system not only detects sirens and screams, but also other potential dangerous environmental sounds. We test our system on three benchmark environment sound classification datasets, separately as well as combined, to perform binary classification between emergency and non-emergency sounds.

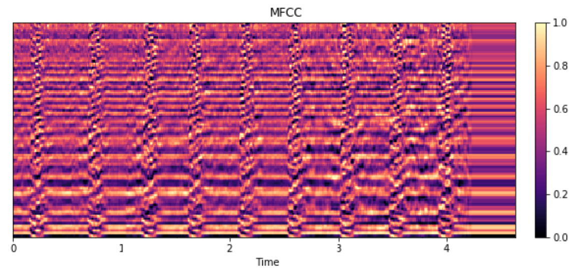
3.1.2.3 Proposed Environment Sound Classification Model

We propose a novel ESC model that consists of multiple feature channels extracted from the audio signal and a new DCNN architecture consisting of separable convolutions, that works on time and frequency domain separately and a depthwise convolution based attention mechanism.
The feature extraction stage consists of four channels of features, which are: Mel-Frequency Cepstral Coefficients (MFCC), Gammatone Frequency Cepstral Coefficients (GFCC), Constant Q-transform (CQT) and Chromagram.
For the classification stage, we propose a CNN architecture that works better for audio data, as shown in Figure 3.7. We use spatially separable convolutions to process time and frequency domain features separately and aggregate them at the end. Also, the downsampling value is different for time and frequency domains in the maxpooling layers. Along side the main DCNN model, we add spatial and channel attention using the depthwise convolution. In the subsequent sub-sections, we explain the feature extraction and classification stages of our model.

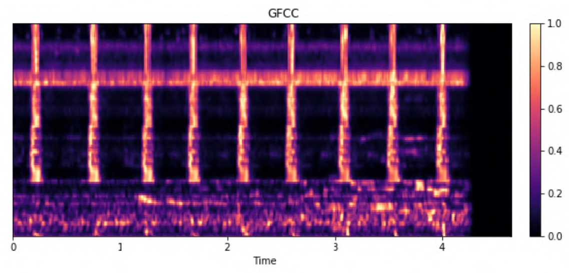
Multiple Feature Channels: Here, we use four major audio feature extraction techniques to create a four channel input for the Deep CNN, namely, Mel-Frequency Cepstral Coefficients (MFCC) [8], Gammatone Frequency Cepstral Coefficients (GFCC) [13], Constant Q-Transform [16] and Chromagram [21]. Incorporating



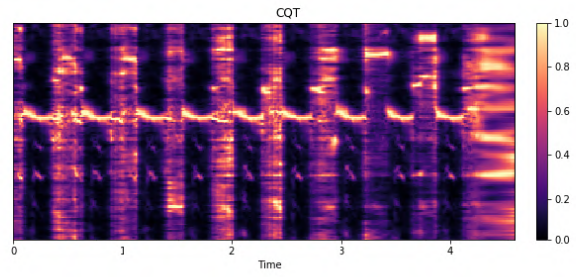
(a) Audio Signals



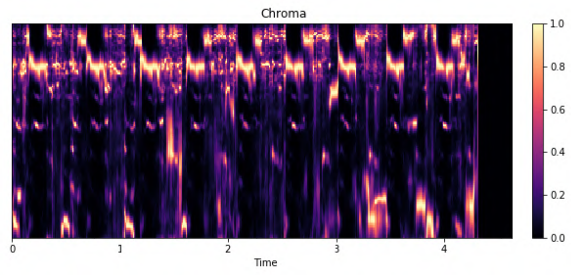
(b) Mel Frequency Cepstral Coefficients



(c) Gammatone Frequency Cepstral Coefficients

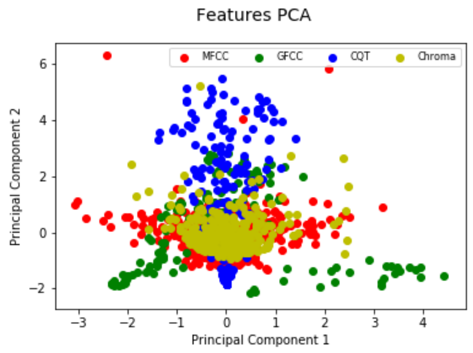


(d) Constant Q-Transform

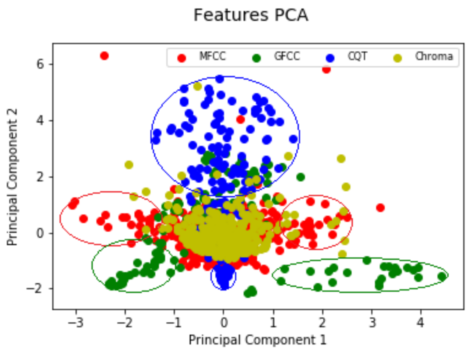


(e) Chromagram

Figure 3.5: Multiple Feature Channels



(a)



(b)

Figure 3.6: PCA of Features

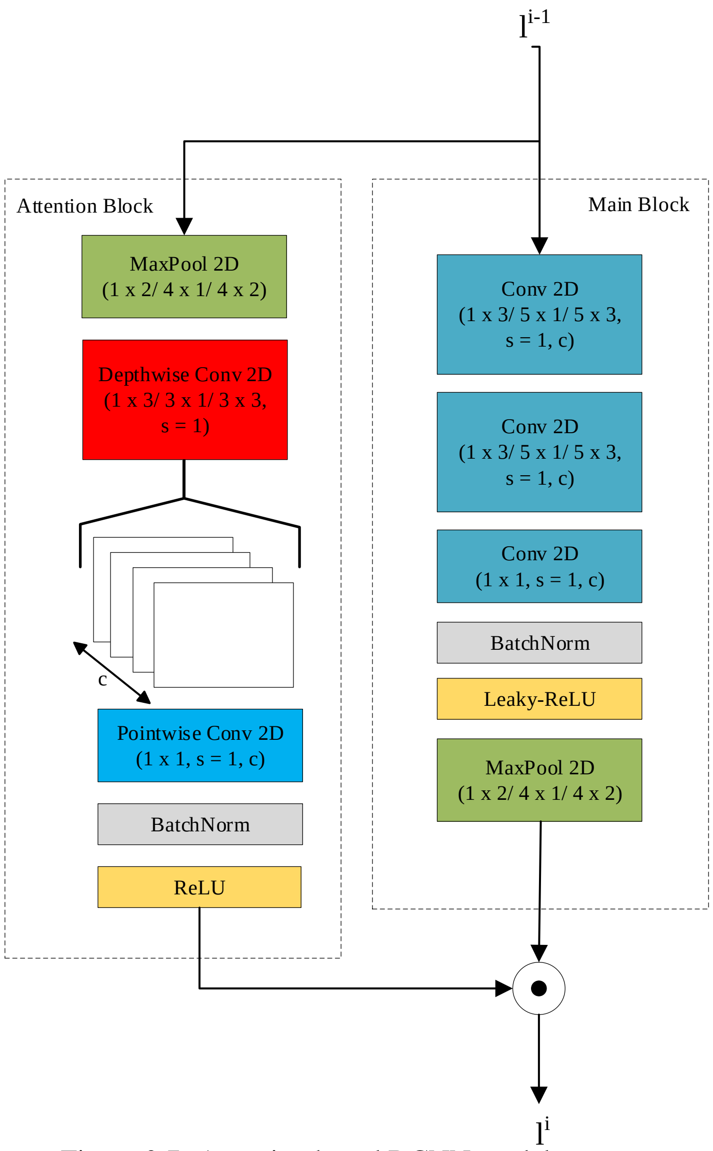
different signal processing techniques that extract different types of information provides the CNN with more distinguishable characteristics and complementary feature representations to accurately classify audio signals.

The MFCC, GFCC, CQT and Chroma features are stacked together to create a four channel input for the Deep CNN. Each feature plays it's part in the classification task. MFCC acts as the backbone by providing rich features, GFCC adds transient sound features, CQT contributes with better low-to-mid frequency range features and finally Chromagram provides pitch category analysis and signal structure information. Figure 3.5 shows a graphical representation of the features extracted from an audio signal (Figure 3.5 a). All features are normalized between 0 and 1 using min-max normalization. From the figure, we can see the contrast in the values of each feature.

Figure 3.6 a shows the Principal Component Analysis (PCA) of the features. We take the first two principal components of the four features we use in our model to create a 2D visualization of the feature space. From the figure we can see that most of features are heavily concentrated in the middle region. But, as shown in Figure 3.6 b, we encircle a few regions that different features provide some amount of different information. Indeed some of these regions might contain irrelevant or outlier information that is not of value to classification. But, as seen in the figure these feature extraction techniques do provide unique and complementary information. Chromagram features provide little distinctive information and shown in the results section, it provides little increase to the performance of the model.

Deep CNN Architecture: Main Block Figure 3.7 shows our proposed Deep CNN architecture for environmental sound classification. The main block consists of five repetitions of *Conv2D-Conv2D-Conv2D-MaxPool-BatchNorm* with different number of kernels and kernel sizes. Almost all convolutional layers are made up of

spatially separable convolutions.
In the case of the ESC task, the input are the features extracted from the audio signals. Each feature set is of the shape (t, f, c), where t is the compressed time domain (compressed due to window size and hop length) and c is the number of channels. Each window of time yields f number of features ($f = 128$ in our model). So, we treat the time domain and the feature domain separately. The kernels with the form $1 \times m$ work on the feature domain and the ones with $n \times 1$ work on the time domain. Using the $1 \times m$ type of convolution operation enables the network to process each set of features from a time window separately. And, the $n \times 1$ type of convolution allows the aggregation of a feature along the time domain. Now, c corresponds to the number of feature extraction methods we adopt (in our model, $c = 4$). So, each kernel works on each channel, which means that all different types of features extracted from the signal feature extraction techniques is aggregated by every kernel. Each kernel can extract different information from an aggregated combination of different feature sets.
Another major advantage of using this type of convolution is the reduction in number of parameters. This is the primary advantage of separable convolutions when they were used in [92] and have probably been used earlier as well.
In case of standard square kernels like $n \times n$, which are used for computer vision tasks, the dimensions of the kernel are in accordance to the image's spatial structure. The 2D structure of an image represents pixels, i.e. both dimensions of an image represent the same homogeneous information. Whereas, in case of audio features, one dimension gives a compact representation of frequency features of a time window and the other dimension represents the flow of time (or sliding time window). So, in order to process information accordingly and respect the information from different dimensions of the input, we use $1 \times m$ and $n \times 1$ separable convolutions.
Deep CNN Architecture: Attention Block We achieve spatial and channel wise attention using a single attention module and dramatically reduce the number of parameters required for attention by using depthwise convolutions. The attention block, shown in Figure 3.7, runs in parallel with a main block. The pooling size and kernel size in the attention block is the same as the pooling and kernel size in the corresponding parallel main block. Using depthwise convolution reduces the number of parameters and thus reduces the overhead of adding attention blocks to the model.
Before the element-wise multiplication of the attention matrix with the main block output, we add a batch normalization layer to normalize the attention weights. Normalization is important for smoothing. The batch-norm layer is followed by a ReLU activation, that makes the attention weight matrix sparse which makes the



l^i-1
Attention Block
MaxPool 2D (1 x 2/ 4 x 1/ 4 x 2)
Depthwise Conv 2D (1 x 3/ 3 x 1/ 3 x 3, s = 1)
c
Pointwise Conv 2D (1 x 1, s = 1, c)
BatchNorm
ReLU
Main Block
Conv 2D (1 x 3/ 5 x 1/ 5 x 3, s = 1, c)
Conv 2D (1 x 3/ 5 x 1/ 5 x 3, s = 1, c)
Conv 2D (1 x 1, s = 1, c)
BatchNorm
Leaky-ReLU
MaxPool 2D (1 x 2/ 4 x 1/ 4 x 2)
l^i

Figure 3.7: Attention based DCNN model

element-wise multiplication computationally efficient.

$$a^i = \phi(BatchNorm(f(MaxPool(l^{i-1})))) \tag{3.1}$$

$$l^i = a^i \odot \hat{l}^i \tag{3.2}$$

Equations 3.1 and 3.2 make up the attention module, where f is the depthwise separable convolution comprising of depthwise and point-wise convolution and ϕ is the ReLU activation function. This single attention module performs both spatial and channel attention. Channel-wise attention requires an attention weight for each output channel of the main block and spatial attention requires an attention weight for each spatial location in the output feature map. Our attention module produces c weights, which enables channel attention, and each weight in c is a matrix of $n \times m$, which enables spatial attention. And, using a single depthwise separable convolution layer we are able to achieve this with considerably less number of parameters and operations.

An advantage of using attention as a separate module that runs in parallel with every main block and connected before and after each main block, with less number of parameters and layers, is that it allows smooth propagation of the gradient like skip or residual connections [122, 123].

3.1.2.4 Proposed Emergency Audio Classification Model

For the task of emergency sound detection, we use our previously proposed environmental sound classification (ESC) model. We adjust the model according to the problem at hand. We reduce the size of the model and remove the attention block considering the task to solve. Our model uses the same multiple feature extraction techniques to create a multi-channel input for the Deep CNN. We apply our model to the problem of emergency sound detection. We treat the task as a binary sound classification problem.

We combine the benchmark ESC datasets into a single big dataset consisting of 10,732 audio files in total. Then, divide the classes into two major categories: Emergency and Non-emergency sounds. Classes that fall under emergency sounds are sirens, fire crackling, glass breaking, gun shot, and thunderstorm. All the other sounds are kept under non-emergency sounds. Since the dataset is highly imbalanced, due to the task at hand, using class weights is important. The dataset consists of just 910 sound files that fall under the emergency sounds category, which is just 8.5% of the total audio duration. Class weights help in boosting the performance of our model.

3.2 Analysis

Emergency analysis requires reliable and accurate information of the environment. In case of fire emergencies, during the emergency analysis stage, gathering information usually does not entail putting the fire personnel in harms way. In [124], the Institute of Medicine, US, gave an extensive review on data collection capabilities and information resources in disaster management. It discusses some interesting guidelines like having a central repository of data, a national emergency information system and improving data collection techniques.

We propose a novel solution to the above problem of emergency analysis. It is not only a data collection technique, but also consists of data analysis. Many such

guidelines and proposals [125] elaborate on efficient data collection and information management, while lacking the crucial step of data analysis.
An analysis of information systems and efficiency in communication during disasters was presented in [126]. It consists of detailed analysis of previous disasters by the Pan-American health organization under the World Health Organization (WHO). It also enlists guidelines for efficient information gathering and communication for emergencies.
There hasn't been much research on using AI and data analytics for the purpose of emergency situation analysis. We concentrate on fire emergencies, which are the most common type of hazard. We focus on extracting and accurately providing insights into the emergency environment in order to help the fire brigade make an informed evacuation plan. For this purpose, we use the computer vision technique of semantic segmentation. We propose to analyse a fire emergency situation by identifying and classifying objects based on their build material to get an idea about their vulnerability to catch fire.
Since there is no dataset that consists of annotation based on build material, we build our own dataset which will be freely available to use. We train state-of-the-art segmentation models like U-Net, SegNet, FCN, PSPNet, DeepLabv3 and DeepLabv3+ on our dataset and compare their results. We also use multitask learning to enable the encoder/backbone to classify between fire and normal images. During inference time, unnecessary computation is avoided, when the decoder part is only activated if a fire is detected by the backbone network. We also show the importance of transfer learning by fine-tuning and testing pretrained PSPNet on our dataset. We also compare different pretrained PSPNet models that are trained on different benchmark datasets.

3.2.1 Semantic Segmentation for Emergency Analysis

We propose semantic scene parsing as an approach to emergency situation analysis. To the best of our knowledge, this is the first time emergency analysis has been performed using image segmentation. Specifically, we consider the case of fire related emergencies. For analysing such situations, we propose to identify and classify objects based on their build material in order to determine their vulnerability to catch fire and thereby obtaining useful information about potential fire spread. For example, a wooden object is more likely to catch fire than a plastic object, which can reveal a potential direction of fire spread.
Along with objects, the fire itself is also segmented. The area covered by the segmented fire can show whether the fire has recently been ignited or is it full-blown.

Since all pixels of a fire are segmented, it could also give information on the direction of the fire, i.e. are the flames bent towards a particular direction (due to wind or any other factor). Apart from fire, smoke is also identified and segmented. Before the fire erupts, smoke provides the first signs of an impending fire hazard. Segmenting smoke could also provide necessary information about the epicentre of fire and its spread direction.
Also, people and other living beings are identified and classified in the environment, which helps in getting head count. Including this information, fire fighters have crucial details about the environment which can result in better evacuation planning. Priority could be given to areas which contain more people for swift evacuation.
Note that the information gained from the analysis stage is readily available to the fire fighters even before they enter the hazardous environment, which not only results in better planning but also reduces the risk for fire fighters themselves, since they enter an unseen and unexplored environment with some important information.
In order to achieve this, we train and test several state-of-the-art segmentation models on our own dataset. We use the DeepLabv3, DeepLabv3+, PSPNet, FCN, SegNet and UNet with multitask learning for better inference computation. To the best of our knowledge, this is the first time semantic segmentation has been used to analyse an emergency situation. The following subsections illustrate the design of our emergency analysis system by describing our fire emergency scene parsing dataset and the segmentation models benchmarked on the dataset.

3.2.1.1 Fire Emergency Scene Parsing Dataset

We uniquely design our dataset consisting of fire and normal images. We use images from our previous fire detection dataset which was used in [102, 103]. We add more images consisting of many different objects in order to properly train the models. The dataset consists of 2065 images which are annotated on the basis of object material, fire and living beings. The fire emergency scene parsing dataset consists of nine categories (+1 for background), which are:

Wood: It is arguably the most vulnerable object during a fire emergency. It can catch fire more easily than other objects and thus spread fire more rapidly.

Plastic: It is one of the less vulnerable objects to catch fire. It can hinder the spread of fire, which makes it an important object material to identify.

Cloth: It is also one of the objects that is vulnerable to catch fire. It is abundant in most environments, which makes it important to identify.

Plants: Trees and vegetation are one of the most vulnerable objects in a fire hazard, which is evident from forest fires. They can be found in indoor as well as outdoor environments. Since forest fires are one of the most common and devastating disasters, it is imperative to identify such objects.

Stone: Identifying and classifying stone objects makes the analysis model generalize well over indoor and outdoor environments. Also, like plastic, usually stone based objects are less vulnerable to catch fire.

Fire: Identifying fire is the central idea behind a fire emergency analysis stage. Segmenting fire can provide information about the nature of fire as well as the magnitude of the fire hazard.

Smoke: One of the first signs of fire, sometimes revealed even before the fire starts, is smoke. It is one of the most important objects to identify since it can act as a signal for an impending fire hazard.

Person: Identifying people and their location is the most important task of the analysis phase. Segmenting people in the environment reveals head count and the distribution of people in the environment.

Other: This object class consists of the objects that can't be identified accurately only through visual data.

Some examples of images with fire from our dataset are shown in Figure 3.8.
However, in our dataset, we also have images displaying normal circumstances as well. It allows the models trained on the dataset to have better generalization performance. For instance, if a fire hazard occurs in an office building, most often only a single room has caught fire. But, all areas of the building must be analysed to identify objects and more importantly people in the building. Also, proper analysis could lead to the discovery of a better evacuation path. So, we include images from many different types of environments like open spaces, offices, homes etc. containing many different kinds of objects, in order to provide varied information to the segmentation models to improve generalization performance. Some examples of such images are shown in Figure 3.9.
To the best of our knowledge, this is the first scene parsing or segmentation dataset that is based on emergency analysis and segmenting objects based on build material. The dataset is imbalanced as the number of instances of each class vary a lot. Keeping this in mind, we also calculate the frequency weighted mean intersection over union metric to compare models.

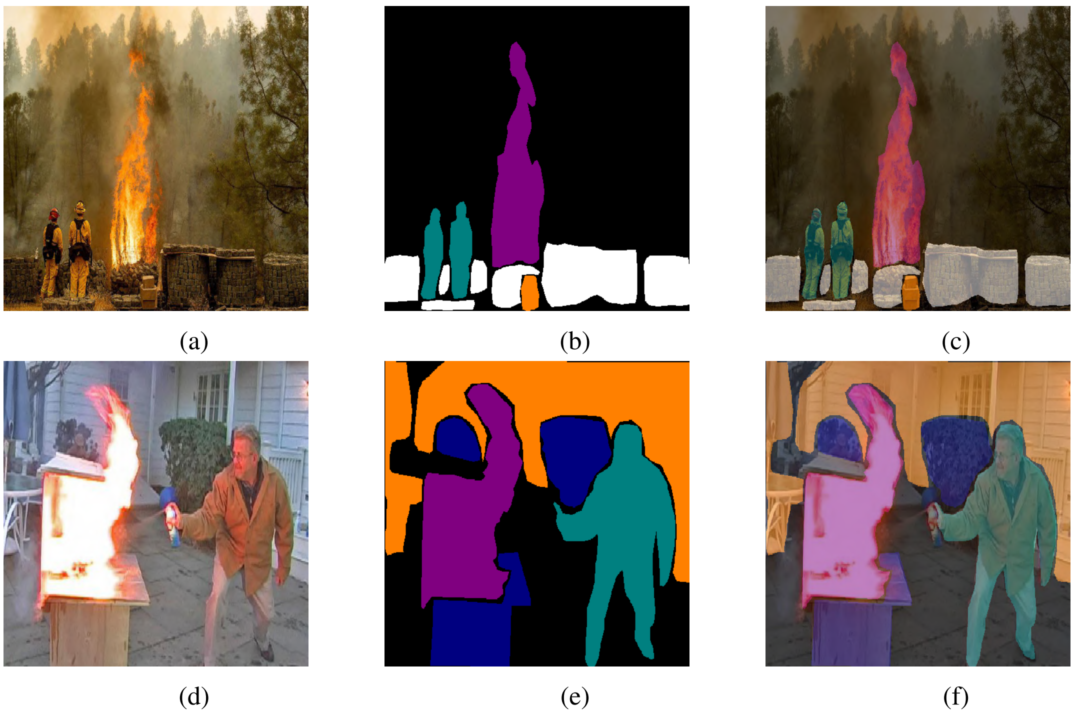


Figure 3.8: Fire Images, Segmentation Masks and Annotated Images

As the results of state-of-the-art segmentation models suggest, this dataset is difficult to classify since objects of the same shape might belong to different classes (different build material) and the shape of fire is usually highly irregular with different colour properties. That's why we choose state-of-the-art segmentation models instead of a small custom model.

3.2.1.2 Semantic Segmentation Models

For the task of fire emergency scene parsing, we employ four state-of-the-art segmentation models. Namely, U-Net [121], SegNet [127], FCN [128], PSPNet [129], DeepLabv3 [130] and DeepLabv3+ [131]. We train these networks from scratch on our dataset. We also fine-tune pretrained versions on the PSPNet to show the effect of transfer learning. Here, we briefly describe each of the segmentation models used in our work.

UNet: U-Net is a Deep CNN that was first developed for medical image segmentation [121]. It was one of the first fully convolutional networks for image segmentation. It consists of contraction and expansion blocks connected via skip connections resulting in a 'U' shaped architecture. The remarkable aspect about U-Net was its ability to achieve good performance by training on very few annotated images. It was also able to segment a 512×512 image in less than a second.

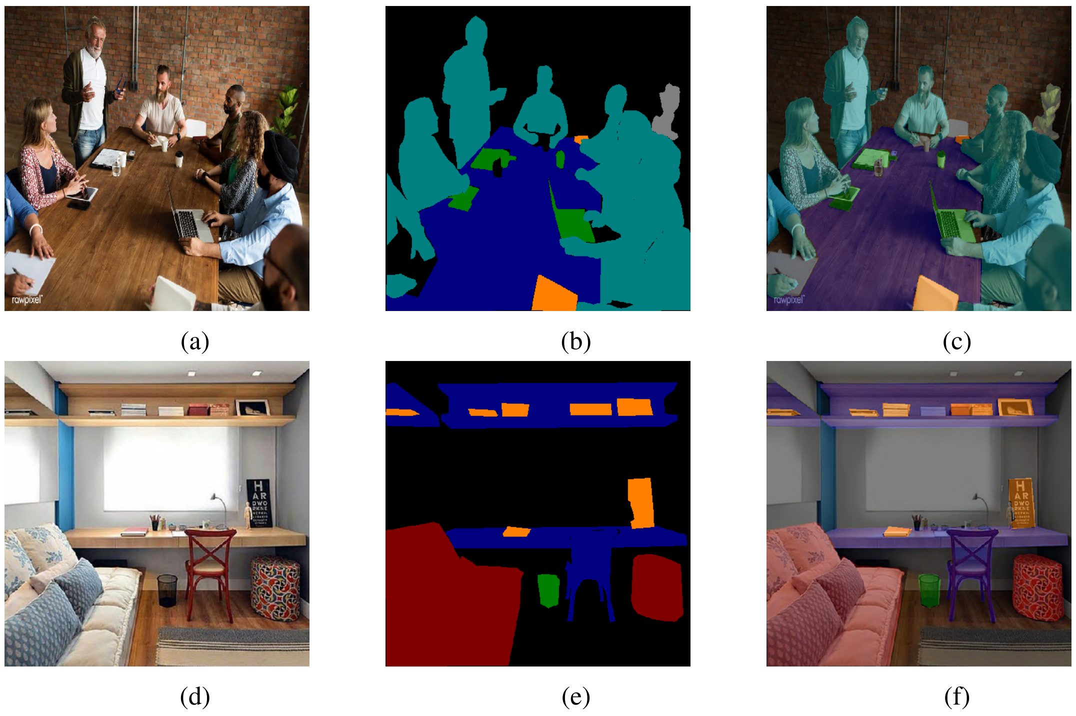


Figure 3.9: Fire Images, Segmentation Masks and Annotated Images

SegNet: Another end-to-end trainable, fully convolutional network for scene parsing and understanding, called SegNet, was proposed in [127]. SegNet consists of an encoder-decoder structure. The VGG16 [30] acts as the encoder to extract features from the input images. The decoder is almost a mirror image of the encoder and is designed to output a pixel-level classification map. The novelty of the model lies in upsampling feature maps using the saved indices of the max-pooled features in the corresponding encoder layers. This produces a sparse 'unpooled' feature map. Convolutional filters are used to produce dense feature maps from these unpooled features. Using unpooling layers dispenses with using trainable upsampling layers. This results in reduced memory consumption and computation.

FCN: In [128], a family of fully connected network architectures for semantic image segmentation was proposed. The adaptation of contemporary classification models like VGG net [30], AlexNet [34], GoogleNet [132] etc. into fully convolutional networks for segmentation was proposed. The dense predictions from classification models were used as image features. Upsampling layers were changed to backward strided convolutions, also called transposed convolutions. In this way, the whole networks consisted of convolutional layers only and hence the name Fully Convolutional Networks. Features from different depths and strides were combined to produce rich segmentation masks. 32, 16 and 8 strided features were combined, pooled and upsampled (using transposed convolutions) to get the final predictions. In this way, the model encompassed coarse as well as fine features to produce accurate predictions.

PSPNet: The PSPNet, proposed in [129], held the state-of-the-art model in the segmentation task in 2017. The main idea behind this model was the Pyramid Pooling module. The features extracted from a backbone network (like the ResNet-50 and ResNet-101 [31] classification models) were used as input to the pyramid pooling module. The module consists of different levels of pooling sizes like 1×1, 2×2, 3×3 and 6×6. These pooled features were followed by a convolutional layer. Finally, the outputs of different levels were upsampled, concatenated and fed to a convolutional classifier to produce the final predictions. The pyramid pooling module enabled the model to capture both local and global context information. The model was trained with deep supervision with auxiliary loss for ResNet-based FCN network.

DeepLabv3 and DeepLabv3+: In 2017, the DeepLabv3 segmentation model, proposed in [130], outperformed the PSPNet to set a new state-of-the-art for semantic segmentation. In [130], the usage of atrous convolutions, also called dilated convolutions, for semantic segmentation was advocated. The main advantage of using atrous convolutions was the increase in the receptive field of the network and it allowed to control the spatial density of feature maps. These atrous convolutions, with different dilation rates, were laid out in a spatial pyramid pooling module, like in the PSPNet, called atrous spatial pyramid pooling (ASPP). The results of these atrous convolutions with different rates were then concatenated to get the final logits. This model was further improved in [131], by proposing a novel decoder module to morph the DeepLabv3 model into an encoder-decoder structure. This improved model is called the DeepLabv3+. The decoder consists of sequential convolutions and upsampling layers with a skip connection from the ASPP module. In order to reduce computation, depthwise separable convolutions were also employed.
All the above models have shown exemplary performance on the image segmentation task. Each one has some advantages and disadvantages over the other models. To see which of these models is best suited for the task of emergency analysis, we train and test all the above models on our fire emergency scene parsing dataset and present our findings in the next chapter.

3.2.1.3 Multitask Learning

In order to avoid unnecessarily executing the decoder part of the segmentation model for every image during inference, we propose to use the encoder/backbone network as a first stage classifier to distinguish between fire and normal images. If a fire is detecting, only then the decoder is activating to segment the image. So, instead of training the backbone network and segmentation decoder separately, we use multitask learning [133] to train the whole model in an end-to-end manner.
This is done by adding a classifier with a sigmoid activation at the end of the backbone network for binary classification between fire and normal images. However, the decoder stage receives the output of the last convolutional feature map of the encoder/backbone network. The end-to-end training of the whole model is performed using a multitask loss which is the combined loss of fire detection and segmentation.
We use per-image binary cross entropy as the classification loss denoted by L_{cls}:

$$L_{cls} = -y\log(p) + (1-y)\log(1-p) \tag{3.3}$$

For the segmentation loss, we use the per-pixel cross entropy, denoted by L_{seg}:

$$L_{seg} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{M} y_{i,c}\log(p_{i,c}) \tag{3.4}$$

where, N is the number of pixels in the image and M are the number of classes. The total loss is simply the sum of the above two losses, denoted by L_{total}:

$$L_{total} = L_{cls} + L_{seg} \tag{3.5}$$

The classification loss is calculated at the end of the encoder/backbone stage, while the segmentation loss is calculated at the end of the decoder stage. The gradients for the segmentation loss flow from the decoder to the encoder as well, but we keep a very small learning rate associated with the segmentation loss for the encoder. So, the task of segmentation depends on the decoder.
Note that, this way of multitask learning to reduce computation is only effective at inference time when the model is deployed in the real world. For unbiased comparison, we run the whole model on all the test images.

3.3 Evacuation

The evacuation stage is probably the most important step in the fire emergency management procedure. This phase entails using relevant information available to carefully plan an effective, efficient and quick response for evacuation from the hazardous environment. We use reinforcement learning (RL) to solve this problem. In order to train an RL agent an environment is required. There are many reinforcement learning libraries that contain simulations and game environments to train reinforcement learning based agents [134–138]. However, currently no realistic learning environment for emergency evacuation has been reported. We build the first realistic fire evacuation environment specifically designed to train reinforcement learning agents for evacuating people in the safest manner in the least number of time-steps possible. The environment has the same structure as OpenAI gym environments, so it can be used easily in the same manner. This environment poses a high level of difficulty, we argue that incorporating the shortest path information (shortest path from each room to the nearest exit) in the DQN model(s) by transfer learning and pretraining the DQN neural network function approximator is necessary. We present a new class of pretrained DQN models called Q-matrix Pretrained Deep Q-Networks (QMP-DQN).

3.3.1 Related Work

Even though this critical application hasn't received adequate attention from AI researchers, there have been some noteworthy contributions. One such paper, focusing on assisting decision making for fire brigades, is described in [139]. Here, the the RoboCup Rescue simulation is used as a fire simulation environment [140]. A SARSA Agent [141] is used with a new learning strategy called Lesson-by-Lesson learning, similar to curriculum learning. Results show that the RL agent is able to perform admirably in the simulator. However, the simulator lacks realistic features like bottlenecks, fire spread and has a grid structure which is too simplistic to model realistic environments. Also, the approach seems unstable and needs information about the state which isn't readily available in real life scenarios.
In [142], multiple coordinated agents are used for forest fire fighting. The paper uses a software platform called Pyrosim which is used to create dynamic forest fire situations. The simulator is mostly used for terrain modeling and a coordinated multiple agent system is used to extinguish fire and not for evacuation.
The evacuation approach described in [143] is similar to the problem we try to solve in this thesis. In [143], a fading memory mechanism is proposed with the intuition

that in dynamic environments less trust should be put on older knowledge for decision making. But arguably, this could be achieved more efficiently by the 'γ' parameter in Q-learning along with prioritized experience replay. Also, the graph based environment used in [143] lacks many key features like fire spread, people in rooms, bottlenecks etc.

The most significant work done on building evacuation using RL is reported in [144]. The evacuation environment is grid based with multiple rooms and fire. The fire spread is modelled accurately and uncertainty taken into account. The multi-agent Q-learning model is shown to work in large spaces as well. Further, the paper demonstrates a simple environment and strategy for evacuation. However, the approach proposed in [144] lacks key features like bottlenecks and actual people in rooms. The grid based environment isn't able to capture details of the building model like room locations and paths connecting rooms.

Some interesting research on evacuation planning take a completely different approach by simulating and modelling human and crowd behaviour under evacuation [145–148]. Our work on evacuation planning is not based on human behaviour modelling or the BDI (Belief-Desire-Intention) framework for emergency scenarios. These methods are beyond the scope of this paper and not discussed here.

3.3.2 Fire Evacuation Environment

We propose the first benchmark environment for fire evacuation to train reinforcement learning agents. To the best of our knowledge, this is the first environment of it's kind. The environment has been specifically designed to simulate realistic fire dynamics and scenarios that frequently arise in real world fire emergencies. We have implemented the environment in the OpenAI gym format [134], to facilitate further research.

The environment has a graph based structure to represent a building model. Let $G = (V, E)$ be an undirected graph, such that $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices that represents n rooms and hallways and $E = \{e_1, e_2, ..., e_m\}$ is a set of edges that represents m paths connecting the rooms and hallways.

To represent the graph consisting of rooms, hallways and connecting paths, we use the adjacency matrix M_A. It is a square matrix consisting of elements $[0, 1]$ that indicate whether a pair of vertices is connected by an edge or not. The adjacency matrix is used to represent the structure of the graph and check the validity of actions performed by the agent. The environment dynamics are defined as follows:

State Each vertex v_i of the graph represents a room and each room is associated with an integer N_i, which is the number of people in that room. The state of the environment is given by a vector consisting of the number of people in each room $S = [N_1, N_2, ..., N_n]$. To force the RL agent to learn the environment dynamics by itself, the environment doesn't provide any other feedback to the agent apart from the state (number of people left in each room) and the reward.

Action An agent performs an action by moving a person from one room to the other and the state is updated after every valid action. Therefore, the action space is discrete. To keep things simple, we restrict the agent to move one person from one room at a time step. The agent can move a person from any room to any other room at any time step, even if the rooms aren't connected to each other by a path. So, the number of possible actions at each step is n^2.
This action space is necessary so that the agent can easily generalize to any graph structure. Also, this enables the agent to directly select which room to take people from and which room to send people to, instead of going through each room in a serial manner or assigning priorities.
When the agent selects an action, where there is no path between the rooms, the agent is heavily penalized. Due to this unrestricted action space and penalization, the agent is able to learn the graph structure (building model) with sufficient training and only performs valid actions at the end. The adjacency matrix is used to check the validity of actions.
Note that our graph based fire evacuation environment has n^2 possible actions (even though many of them are illegal moves and incur huge penalties), where n is the number of rooms. Even for a small toy example of $n = 5$ rooms, the total number of possible actions is 25, which is a lot more than almost all of the OpenAI gym environments and Atari game environments [134].

Reward We design a reward function specifically suited for our environment. We use an exponential decay function to reward/penalize the agent depending on the action it takes and to simulate fire spread as well. The reward function looks like this:

$$r(v_j, t) = -[d(v_j, t)]^t \tag{3.6}$$

where, t is the time step, v_j is the room where a person is moved to and $d(.)$ is the degree of fire spread for a room. $d(.)$ returns a positive number and if a room has a higher value of degree of fire spread, that means that fire is spreading more rapidly towards that room. We explicitly assign degrees to each room using a degree vector $D = [d(v_1, t), d(v_2, t), ..., d(v_n, t)]$, where the maximum value belongs to the room where the fire is located.
Using such a reward function ensures the following: Firstly, the reward values drop exponentially every time step as the fire increases and spreads. Secondly, the reward

of an action depends on the room where a person is moved to. The reward function will penalize an action more heavily if a person is moved to a more dangerous room (higher degree of fire spread towards that room). This is because the function yields more rapidly decaying negative rewards. Lastly, the function yields a negative reward for every action which forces the agent to seek the least number of time-steps. The reward for reaching the exit is a constant $[r(v_j = exit, t) = +10]$.

Fire Location(s) and Exit(s) The room where the fire occurs is given the highest degree, hence the maximum penalty for entering. The direction of fire spread is randomly decided and the degrees are assigned accordingly. The degrees are updated gradually to simulate fire spread.

$$d(v_j, t+1) = d(v_j, t) + \delta_j; \quad \forall v_j \quad in \quad V \tag{3.7}$$

where, δ_j is a small number $(0 \leq \delta \leq 1)$ associated with v_j. δ is assigned to each room according to fire spread direction. So, δ can be used to determine fire spread direction, since higher value of δ for a room means that fire is spreading towards that room more rapidly.
The exit is also treated like a room. The only difference being that the agent gets a positive reward for moving people to the exit. The number of people at the exit is reset to zero after every action. The rooms which are exits are stored in a vector $\mathcal{E}$.

Bottleneck Probably one of the most important feature in our proposed fire evacuation environment that enhances realism is the bottlenecks in rooms. We put an upper limit on the number of people that can be in a room at a time step. This restriction ensures congestion control during evacuation, which has been a huge problem in emergency situations. The bottleneck information is not explicitly provided to the agent, instead the agent learns about this restriction during training, since a negative reward is received by the agent if the number of people in a room exceed the bottleneck value. The bottleneck $\mathcal{B}$ is set to 10 in our experiments.

Uncertainty To take into account uncertain behaviour of the crowd and introduce stochasticity in the environment, a person moves from one room to the other with probability $1-p$. This means that an action a_t, selected by the agent at time-step t, is performed with probability $1-p$ or ignored with probability p. If the action is ignored, then there is no change in the state, but the reward received by the agent is as if the action was performed. This acts like a regularizing parameter and due to this, the agent is never able to converge to the actual global minimum. In our experiments, the uncertainty probability p is kept at 0.1.

Terminal Condition The terminal/goal is reached once there are no people in any of the rooms [$\sum_{i=1}^{n} N_i = 0$].

Pretraining Environment: We create two instances of our environment: one for fire evacuation and the other for shortest path pretraining. For the pretraining instance, we consider only the graph structure and the aim is to get to the exit from every room in the minimum number of time-steps.
The pretraining environment consists of the graph structure only, i.e. the adjacency matrix M_A. The pretraining environment doesn't contain fire, the number of people in each room or bottlenecks. The rewards are static integers: -1 for every path to force the agent to take minimum time-steps, -10 for illegal actions (where there is no path) and +1 for reaching the exit. The agent is thus trained to incorporate shortest path information of the building model.

3.3.3 Q-matrix Pretrained Deep Q-Networks

For the proposed graph based fire evacuation environment, we also present a new reinforcement learning technique based on the combination of Q-learning and DQN (and its variants). We apply tabular Q-learning to the simpler pretraining environment, with a small state space, to learn the shortest paths from each room to the nearest exit. The output of this stage is an $n \times n$ Q-matrix which contains q-values for state-action pairs according to the shortest path.
This Q-matrix is used to transfer the shortest path information to the DQN agent(s). This is done by pretraining the agent's neural network by deliberately overfitting it to the Q-matrix. After pretraining, the neural network weights have the shortest path information incorporated in them. Now, the agent is trained on the complete fire evacuation environment to learn to produce the optimal evacuation plan.
The main purpose of using such a strategy of training an agent by pretraining it first is to provide the agent with vital information about the environment beforehand, so that it doesn't have to learn all the complexities of the environment altogether. Since, after pretraining, the agent knows the shortest paths to the nearest exits in the building, dealing with other aspects of the environment like fire, fire spread, number of people and bottlenecks is made easier.
We provide two instances of our environment: simpler shortest path pretraining instance and complex fire evacuation instance. First, the agent is pretrained on the simpler instance of the environment (for shortest path pretraining) and then trained on the more complex instance (for optimal evacuation). This approach of training the agent on a simpler version of the problem before training it on the actual complex problem is somewhat similar to curriculum learning [149].

We also add a small amount of noise or offset to the Q-matrix produced by training on the pretraining environment instance. This is done by adding or subtracting (depending on the q-value) a small σ to each element of the Q-matrix.

$$Q(s,a) = \begin{cases} Q(s,a) + \sigma, & \text{if } Q(s,a) \leq 0 \\ Q(s,a) - \sigma, & \text{if } Q(s,a) > 0 \end{cases}$$

where, σ can be thought of as a regularization parameter, which is set to 10 in our experiments. Adding noise to the Q-matrix is necessary because we don't want the DQN agent to just memorize all the paths and get stuck at a local minimum. The actual fire evacuation instance is complex, dynamic and has uncertainty which means that an optimal path at time-step t might not be the optimal path at time-step $t + k$. The hyperparameter σ acts as a regularizer.
Note that we add σ if the element of the Q-matrix is negative or zero and subtract σ if the element is positive. This is done to offset the imbalance between good and bad actions. If we just add or subtract σ then the relative difference between q-values would remain the same. Conditional addition or subtraction truly avoids the DQN based agent from being biased to a particular set of actions leading to an exit.

3.3.3.1 Convergence

The paper [150] thoroughly analyses and proves conditions where task transfer Q-learning will work. We use the proved propositions and theorems from [150] to show that pretraining works in our case. It states that, if the distance between two MDPs is small, then the learned Q-function from the pretraining task is closer to the optimal of the fire evacuation task compared to random initializations and hence helps in convergence to an optimum and improves the speed of convergence. For our case, this seems obvious since the two MDPs are instances of the same MDP.
Now, to prove that our method has guaranteed convergence, we need to prove that the Q-matrix is able to capture the shortest path information accurately. The guarantee of convergence for Q-learning has been discussed and proved in many different ways and for general as well as unique settings [27, 151]. The convergence of Q-learning is guaranteed, while using the update rule given in equation 3.8, if the learning rate η is bounded between $0 \leq \eta < 1$ and the following conditions hold:

$$\sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} [\eta_t]^2 < \infty \tag{3.8}$$

Then, $Q_t(s, a) \longrightarrow Q^*(s, a)$ as $t \longrightarrow \infty, \forall s, a$, with probability 1. This means that for the learning rate conditions to hold with the constraint $0 \leq \eta < 1$, all state-action pairs must be visited an infinite number of times. Here, the only complication is that some state-action pairs might never be visited. We run Q-learning on the pretraining environment for ~ 1000 episodes so ensure that the Q-matrix converges to $Q^*(s, a)$.

3.3.4 Scalability: Large and Complex Real World Scenario - University of Agder Building

To prove that our method is capable of performing on large and complex building models, we simulate a real world building, i.e., the University of Agder A, B, C and D blocks, and perform evacuation in case of fire in any random room(s). This task is especially difficult because of the resulting complex graph structure of the building and the large discrete action space. We consider the A, B, C and D blocks which are in the same building. The total number of rooms in this case is $n = 91$, which means that the number of all possible actions is 8281. This discrete action space is many times larger than any other OpenAI gym environment or Atari game environments [134]. Even the Go game has $19 \times 19 + 1$, i.e., 362 actions.
We propose a simple approach to deal with large number of actions. Our method consists of two stages: One-Step Simulation (OSS) of all actions resulting in an action importance vector A_I and then element-wise addition with the DQN output for training.

3.3.4.1 One-Step Simulation and Action Importance Vector

We make use of the pretraining environment instance to calculate the action importance vector A_I. We simulate all possible actions for each state/room for one time-step in the pretraining environment. All rewards received for these actions taken from room s and returns the k best actions for each room s which yield the k highest rewards.
The k best actions returned by one-step simulation for all rooms s are accumulated into a single vector of actions.
After we have a unique index for all selected actions in the environment, we form the action importance vector A_I by placing 0 at index l, if the l^{th} action is present in the vector x, which consists of all the k best actions for each room s, otherwise, a large negative number (like -9999) at index l.
The action importance vector can be though of as a fixed weight vector which contains weight 0 for good actions and a large negative weight for others. A_I is then

added element-wise with the output of the DQN $\hat{Q}$ to produce the final output Q^* on which the DQN is trained on.

$$Q^* = \hat{Q} \oplus A_I \tag{3.9}$$

This makes the Q-values of the good actions to remain the same and reduces the Q-values of other actions to huge negative numbers. This method effectively reduces the action space from $O(N^2)$ to $O(kN)$, where $k \ll N$. Hence, our complete method consists of shortest path pretraining using Q-matrix transfer learning and action space reduction by one-step simulation and action importance and finally DQN based model training and execution. The shortest path pretraining provides the model with global graph connectivity information and the one-step simulation and action importance delivers local action selection information.
The action importance vector can also be thought of as an attention mechanism [35, 152–154]. Most of the attention mechanisms employ a neural network or any other technique to output an attention vector which is then combined with the input or an intermediate output to convey attention information to a model. Unlike these methods, our proposed model combines the action importance vector with the output of the DQN. This means that the current action selection is based on a combination of the Q-values produced by the DQN and the action importance vector, but the training of the DQN is impacted by the attention vector in the next iteration of training, as the final output of the i^{th} iteration is used as the label for training the model at the $i+1^{th}$ iteration.
One major advantage of such an attention mechanism used in our method is that, since the graph based environment has a fixed structure, the attention vector needs to be calculated just once at the beginning. We test our method on the University of Agder (UiA), Campus Grimstad building with blocks A, B, C and D consisting of 91 rooms.
Note that, unlike the usual attention based models, we do not perform element-wise multiplication of the attention vector with the output of a layer. Instead, we add the attention vector because initially the DQN model will explore the environment and will have negative Q-values for almost all actions (if not all). This means that if we use a vector of ones and zeros for good and bad actions respectively and multiply element-wise with the output of a layer then, the Q-values of good actions will be copied as it is and the Q-value of other actions will become zero.
If the Q-value of good actions is negative in the beginning due to exploration (and lack of learning since it is the beginning of training), then the max function in the Q-value selection equation will select bad actions since they are zeros and good

actions are negative. This will lead to catastrophic behaviour of the system and it will never converge. So, instead we use addition with zeros for good actions so that they remain the same and with large negative numbers for other actions so that their Q-values become so low that they are never selected.

3.4 Summary

In this chapter, we have presented our contributions in detail. We presented methods for each stage of the fire emergency management pipeline. For visual detection, we used state-of-the-art deep convolutional neural networks like VGG16 and ResNet50 for detecting fire in images. We then improved upon this approach by building a hybrid model consisting of VGG16/ResNet50, acting as feature extractors, and replacing the fully connected layers with Extreme Learning Machine classifier. This improved performance and greatly increased the speed of convergence and inference, while reducing the number of parameters.
On the other hand, for audio detection, we proposed a novel environment sound classification model consisting of multiple signal feature channels, deep convolutional neural networks with separable convolutions for time and feature domain and an attention block. This model was able to achieve state-of-the-art performance on benchmark ESC datasets. We used a lighter version of this model for performing binary classification between emergency and non-emergency sounds.
In the analysis phase, the main concern was to get relevant and necessary information without putting fire emergency services in harms way. So, we proposed to use semantic segmentation to identify inflammable objects and people in the emergency environment. We applied state-of-the-art segmentation models like the U-Net, SegNet, FCN and PSPNet, on our own fire scene parsing dataset consisting of 10 segmentation classes.
For the final and most important phase of fire emergency management, i.e. evacuation, we used reinforcement learning to train an agent to evacuate all people in a building from any location to the nearest exit in the shortest possible time while avoiding any hazardous situations. To train such an agent we built our own simulator based on the OpenAI gym framework. The simulator consists of realistic features such as fire spread, bottleneck, dynamic and random behaviour, and uncertainty in action. We also proposed a new method that was capable of learning in such a complex environment. We call it Q-matrix pretrained DQN, which first uses Q-learning to learn the shortest path from every room to the nearest exit and incorporates this information in a DQN based agent by transfer learning. Then, the

pretrained DQN is trained on the fire evacuation environment.
We also show that this method can scale well with a few minor additions. We test this method on a large scale fire evacuation environment consisting of 91 rooms, resulting in 8281 actions. In order to reduce the action space, we use one-step simulation on all nodes to obtain the k best actions and create the action importance vector. We replace those k actions with zero and set the rest of the actions to a large negative number. This vector is then added to the final output of the DQN agent, so that it weighs bad actions with negative values and good actions remain unchanged. In this way, the action space is reduced drastically and the Q-matrix pretrained DQN agent is able to learn near optimal evacuation paths in a large scale realistic fire evacuation environment.
All these contributions give solutions to the research questions formulated in Chapter 1 Section 1.2. Hence, we provide an AI solution to each stage of the fire emergency management process that outperforms existing techniques and in some cases sets up a completely new paradigm.

Chapter 4

Experimental Evaluation and Results

This chapter provides details of the exhaustive experimentation performed on the proposed models and displays the results. We present the experimentation methodology for each contribution and subsequently show the results. This chapter is organized in the same manner as Chapter 3, where we give a detailed account of our contributions. Here, we show the results obtained for each of our contributions and the setup of our experiments.

4.1 Detection

Here, we show the experiments and results of the contributions proposed in Section 3.1. We show the results of our models for both visual and audio detection.

4.1.1 Visual Detection

We proposed two ways of detecting fire in images using Deep CNNs as presented in Sections 3.1.1.4 and 3.1.1.5. First, we used state-of-the-art Deep CNNs, VGG16 and ResNet50 to detect fire in images. Then, we further improve performance and reduce training and inference time of these models by removing the fully connected layers and replacing them with the ELM classifier. We use our own fire dataset, details to which can be found in Section 3.1.1.3.

4.1.1.1 Deep Convolutional Neural Networks for Fire Detection

Experimental Setup: We conducted our experiments to compare training and testing accuracies and execution times of the VGG16 and Resnet50 models including modifications. We also trained a simple CNN which is used in [66] and compare with much deeper models to show why deeper and more complex models are necessary for fire detection on our dataset. We also train the modified VGG16 and Resnet50 models and compare the performance. We used pre-trained Keras [155] models and fine-tuned the fully-connected layers on our dataset. The training of the models was done on the following hardware specifications: Intel i5 2.5GHz, 8GB RAM and Nvidia Geforce GTX 820 2GB GPU. Each model was trained on the dataset for 10 training epochs with the ADAM optimizer [54] with default parameters $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

Results: Table 4.1 shows the results of our experiments. The simple CNN model labels all images as non-fire which means that it is unable to extract relevant features from the dataset and cannot handle unbalanced datasets, which we can see from the training accuracy which is exactly equal to the percentage of non-fire images in the training set. So, the simple CNN overfits on the majority class of the unbalanced training dataset. Since, the training and fine-tuning methods for all models used here are the same, at the end it comes down to the architecture of the model. This justifies the use of deeper models like VGG16 and Resnet50. The simple CNN tested on our dataset is similar to the one used in [66].

The deep models achieve testing accuracy greater than 90%. And, the modified

Table 4.1: Comparison between CNN models

Model	Training accuracy	Training time (in sec)	Testing accuracy	Testing time (in sec)
VGG16	100	7149	90.19	121
VGG16 (modified)	100	7320	91.18	122
Resnet50	100	15995	91.18	105
Resnet50 (modified)	100	16098	92.15	107
Simple CNN [66]	89.25	112	50.00	2

VGG16 and Resnet50 models outperform the base models by a small margin with slightly higher training time. It seems obvious that adding fully-connected layers to a network would increase accuracy. But on such a small dataset, the trade-off between accuracy and training time is quite poor, so we stop after adding just one fully connected layer. We also tested for more fully-connected layers (which is feasible since the model is pre-trained) but the increase in accuracy compared to increase in training time was too small.

Overall, the deep models perform well on this dataset. This shows that these models generalize well even when there is lack of training data. This means that if we want to slightly alter what the model does, we do not require large amount of data for retraining.

4.1.1.2 The Hybrid Model: Deep CNN-ELM for Fire Detection

Experimental Setup: The experimental setup is the same as 4.1.1.1.

Results: Our ELM hybrid models are tested on our dataset and compared with SVM hybrid models and the original VGG16 and Resnet50 Deep CNN models. Table 4.2 and Table 4.3 show the results of the experiments. The dataset was randomly split into training and testing sets. Two cases were considered depending on the amount of training data. The Deep CNN models (VGG16 and Resnet50) were trained only on 80% training data, since 60% is too less for these models. All the hybrid models have been trained on both 60% and 80% of training data.

One point to be noted here is that, the SVM hybrid models contain an additional

Table 4.2: Accuracy and Execution time

Model	D_T	Acc_{train}	T_{train}	T^C_{train}	Acc_{test}	T_{test}
VGG16 (pre-trained)	80	100	7149	6089	90.19	121
VGG16 (modified)	80	100	7320	6260	91.176	122
Resnet50 (pre-trained)	80	100	15995	13916	91.176	105
Resnet50 (modified)	80	100	16098	13919	92.15	107
VGG16+SVM	60	99.6	2411	1352	87.4	89
VGG16+SVM	80	100	2843	1784	93.9	81
VGG16+ELM	60	100	1340	281	93.9	24
VGG16+ELM	80	100	1356	297	96.15	21
Resnet50+SVM	60	100	3524	1345	88.7	97
Resnet50+SVM	80	100	4039	1860	94.6	86
Resnet50+ELM	60	100	2430	251	98.9	32
Resnet50+ELM	80	100	2452	272	99.2	26

D_T is the percentage of total data used for training the models.
Acc_{train} and Acc_{test} are the training and testing accuracies respectively.
T_{train} and T_{test} are the training and testing times for the models.
T^C_{train} is the time required to train the classifier part of the models.

fully-connected layer of 4096 neurons, while the ELM is directly connected to the last pooling layer. The results in Table 4.2 show that the ELM hybrid models outperform the VGG16, Resnet50 and SVM hybrid models by achieving higher accuracy and learning much faster. In general, we can see that the hybrid models

outperform the state-of-the-art Deep CNNs in terms of both accuracy and training time.
Apart from accuracy and training time, another important point drawn from the results is the amount of training data required. As we already know, Deep Neural Networks (DNN) require huge amount of training data. So, using pre-trained models can be highly beneficial, as we only need to fine-tune the fully-connected layers. But, with models like VGG16 and Resnet50 which have large fully-connected layers, even fine-tuning requires large amount of training data. We had to train the VGG16 and Resnet50 on at least 80% training data otherwise they were overfitting on the majority class, resulting in 50% accuracy.
But in case of hybrid models, especially ELM hybrid models, the amount of training data required is much less. Even after being trained on 60% training data, the ELM models were able to outperform the original VGG16 and Resnet50 models which were trained on 80% training data. This shows that reducing the fully-connected layers, or replacing them with a better classifier can reduce the amount of training data required. Also, the ELM is more robust towards lack of training data which adds to this advantage.
Among the hybrid models, the ELM hybrid models outperform the SVM hybrid models both in terms of testing accuracy and training time. Also, we can see that the hybrid models with Resnet50 as the feature extractor achieves better results than the hybrid models with VGG16 as the feature extractor. This is due to the depth and the residual connections in Resnet50 in contrast to the simple, shallower (compared to Resnet50) and sequential nature of VGG16.
Table 4.3 compares results between different number of hidden neurons used by

Table 4.3: Number of Hidden Neurons in ELM

CNN Features	**# hidden neurons**	**Testing accuracy**
VGG16 Feature Extractor	4096	93.9
VGG16 Feature Extractor	8192	94.2
VGG16 Feature Extractor	16384	91.1 (Overfitting)
Resnet50 Feature Extractor	4096	98.9
Resnet50 Feature Extractor	8192	99.2
Resnet50 Feature Extractor	16384	96.9 (Overfitting)

ELM. The accuracy increases as the number of hidden neurons increase. The models are tested for 2^{12}, 2^{13} and 2^{14} number of neurons. The testing accuracy starts to decrease for 2^{14} neurons, which means the model overfits. All the tests in Table 4.3 were conducted with 60% training data.

4.1.2 Audio Detection

We propose general purpose emergency detection system using audio signals that can be used to complement visual detection. For this task, we first propose a novel Environment Sound Classification (ESC) model that sets a new state-of-the-art on benchmark ESC datasets as discussed in Section 3.1.2.3. We use a reduced version of this model to perform emergency detection using sound. The benchmark datasets used for the ESC task are ESC-10, ESC-50 [156] and UrbanSound8K [157]. For the emergency detection task, we simply combine these datasets and assign the classes sirens, fire crackling, glass breaking, gun shot, and thunderstorm to the emergency class and the rest to the non-emergency class.

4.1.2.1 Proposed Environment Sound Classification Model

Experimental Setup: We report state-of-the-art results on ESC benchmark datasets, i.e. UrbanSound8K, ESC-10 and ESC-50, using the proposed model. The ESC-10 and ESC-50 contain 2000 audio files of 5 seconds length each, while UrbanSound8K consists of 8732 audio files of 4 seconds each. ESC-10 and UrbanSound8K contain audio from 10 classes while ESC-50 has 50 classes. We use k-fold cross-validation on the specified folds and report the average accuracy across the folds. For ESC-10 and ESC-50, $k = 5$ and for UrbanSound8K, $k = 10$. We use Tensorflow and Keras to implement our CNN classifier and Librosa [158] and the Matlab Signal Processing Toolbox [159] for audio processing and feature extraction. In terms of hardware, we use the NVIDIA DGX-2 consisting of 16 NVIDIA Tesla $V100$ GPUs with 32 Gigabytes of VRAM each and a system memory of 1.5 Terabytes.
For every feature extraction technique, we extract 128 features for each window of length 1024 (3.2 ms) with a hop length of 512 (1.6 ms) at 32kHz. We normalize all feature vectors using min-max normalization. Our DCNN model is trained to minimize the categorical cross-entropy loss using the Adam optimizer with Nestorov momentum, along with Dropout of 0.25 after the dense layer. and weight decay of $\lambda = 0.1$. We run our model for 500 epochs per fold. We set the initial learning rate of training to 0.01 and decrease it by a factor of 10 every 150 epochs. As shown in [98], mix-up data augmentation plays a very important role in improving performance, especially when the model is large and data is scarce. We use a mini-batch size of 200. Table 4.4 displays the results of previous state-of-the-art ESC models that tested their methods on one or more of the three benchmark datasets. All of these models have been briefly described in Section 3.1.2.1. The last row of the table shows the results of our proposed model on the three datasets.

Table 4.4: Previous state-of-the-art ESC models vs Proposed model

Model	ESC-10	ESC-50	US8K
EnvNetv2+strong augment [96]	91.30	84.70	78.30
PiczakCNN [94]	90.20	64.50	73.70
CNN+Mixup [98]	91.70	83.90	83.70
FBEs$\oplus$ConvRBM-BANK [97]	-	86.50	-
CRNN+channel & temporal Attention [100]	94.20	86.50	-
Multi-stream+temporal Attention [101]	94.20	84.00	-
TSCNN-DS [99]	-	-	97.20
Multiple Feature Channel + Deep CNN with Attention (Proposed)	**94.75**	**87.45**	**97.52**

Results: We show the advantages of using multiple features, data augmentation, depthwise convolutions and attention mechanism from our experiments on the three benchmark datasets. Table 4.9 containing the results of our experiments with different combination of features and the effect of data augmentation.
Using separable convolutions (spatial or depthwise), has the advantage of reducing the number of parameters in the model. We use spatially separable convolutions in our main block and depthwise separable convolutions in the attention block. In Table 4.5, we show the effect of using separable convolutions in terms of the number of parameters and model performance. The DCNN-5 is the model without attention and DCNN-5 SC is with standard convolutions instead of separable convolutions. The separable convolutions, 1×3 and 5×1, is replaced by 5×3 convolution operation. We use padding when necessary to keep the model depth valid according to the input, since standard rectangular convolutions reduce the output dimensions more quickly.
From Table 4.5, we can see that, for the task of environment sound classification, the spatially separable convolutions have less number of parameters and perform better than standard convolutions. DCNN-5 SC has 130K more parameters than DCNN-5 and obtains 3.25% lower accuracy than DCNN-5 on the ESC-50. Adding the attention mechanism just adds 20K more parameters and increases the performance by 2.7%, courtesy of depthwise convolutions. Using standard convolutions to build the attention model results in an increase of 90K parameters and 0.4% accuracy.
These findings are consistent with the UrbanSound8K dataset. The difference in the number of parameters between the datasets for the same models is because of the

Table 4.5: Performance Comparison of Number of Parameters on ESC-50 and UrbanSound8K

Model	**Parameters ESC-50**	**ESC-50**	**Parameters US8K**	**US8K**
DCNN-5	1.27M	84.75	0.87M	94.25
ADCNN-5	**1.29M**	**87.45**	**0.89M**	**97.52**
DCNN-5 SC	1.40M	81.50	1.04M	91.25
ADCNN-5 (without Depthwise Sep. Conv.)	1.36M	87.05	0.97M	96.35

Table 4.6: Performance of different number of feature coefficients on ESC-50 and UrbanSound8K

Model	**# Features**	**ESC-50**	**US8K**
	48	80.12	89.25
ADCNN-5	64	85.25	94.25
	96	86.15	95.50
	128	**87.45**	**97.52**

difference in input shapes. UrbanSound8K has 4 seconds long audio files, whereas, ESC-50 has 5 seconds long. So, both of them sampled at 32kHz produce different number of time windows. The input shape for ESC-50 is $\langle 313, 128, 4\rangle$ and for UrbanSound8K is $\langle 250, 128, 4\rangle$ represented as $\langle$time-windows, features, channels$\rangle$.

We also test our model with fewer number of features extracted by the audio feature extraction methods. Table 4.6 shows the results when the number of features are reduced. The model accuracy monotonically increases with the increase in the number of features. We stop at 128 features, which produces the best results, to avoid increasing the complexity of the model.

The same tests were conducted on the ESC-10 dataset. The results were consistent with the findings shown above. ESC-10 is a subset of the ESC-50 dataset. We also report state-of-the-art performance on the ESC-10 dataset with 94.75% accuracy.

4.1.2.2 Proposed Emergency Audio Classification Model

Experimental Setup: To test our model for this task we consider two benchmark environmental sound datasets, the ESC-50 [156] and UrbanSound8K [157]. We combine these datasets into a single big dataset consisting of 10,732 audio files in total. Then, divide the classes into two major categories: Emergency and Non-emergency sounds. Classes that fall under emergency sounds are sirens, fire crackling, glass breaking, gun shot, and thunderstorm. All the other sounds are kept

under non-emergency sounds.
The implementation and setup is similar to 4.1.2.1The loss is calculated using class weights for each training instance. Class weights are not used for testing. We use k-fold cross validation, with $k = 5$ and present the average test accuracy of the folds as the final performance metric.

Table 4.7: Effect of Class weights

Model	Class weights	Accuracy
DCNN-5	No	94.09
DCNN-5	Yes	**99.56**

Results: Table 4.7 shows the effect of class weights on the performance of the model. Since the dataset is highly imbalanced, due to the task at hand, using class weights is important. The dataset consists of just 910 sound files that fall under the emergency sounds category, which is just 8.5% of the total audio duration. Due to this, even though the accuracy of the model without class weights is above 90%, still it does not perform well on this task. It overfits on the non-emergency class which has 91.5% of the total instances.
We also test different architectures based on size of the Deep CNN as shown in Table 4.8. All the models in table are trained with class weights. DCNN-6 and DCNN-7 tend to overfit the training data while DCNN-3 and DCNN-4 does not contain enough layers/parameters to learn the data patterns. Therefore, DCNN-5 is the best performing architecture. Hence, our model with five repetitions of *Conv2D-Conv2D-MaxPool-BatchNorm* and four channel input consisting of MFCC, GFCC, CQT and Chromagram features is able to achieve 99.56% accuracy for the task of emergency sound detection on the combination of ESC-50 and UrbanSound8K datasets.

Table 4.8: Deep CNN Architecture Performance

Model	No. of Layers	Accuracy
DCNN-3	15	92.49
DCNN-4	19	95.25
DCNN-5	**23**	**99.56**
DCNN-6	27	99.25
DCNN-7	31	98.45

Table 4.9: Performance on the ESC-50 and UrbanSound8K
Impact of combinations of different feature sets with and without augmentation

Model	DA*	MFCC	GFCC	CQT	Chroma	ESC-50	US8K
		✓				80.25	88.45
			✓			78.15	87.12
				✓		79.92	87.75
					✓	72.45	82.20
		✓	✓			83.25	92.50
		✓		✓		83.50	94.35
		✓			✓	81.75	89.65
ADCNN-5	Yes		✓	✓		81.52	90.86
			✓		✓	79.95	88.56
				✓	✓	80.65	89.75
		✓	✓	✓		86.05	96.85
		✓	✓		✓	84.35	94.90
		✓		✓	✓	85.65	95.55
			✓	✓	✓	84.15	93.25
		✓	✓	✓	✓	**87.45**	**97.52**
		✓				75.65	85.50
			✓			73.50	83.85
				✓		74.25	84.75
					✓	69.35	79.25
		✓	✓			79.12	88.25
		✓		✓		78.55	89.50
		✓			✓	76.30	86.15
ADCNN-5	No		✓	✓		77.62	86.32
			✓		✓	74.75	84.45
				✓	✓	75.15	85.22
		✓	✓	✓		82.95	92.90
		✓	✓		✓	82.22	89.85
		✓		✓	✓	83.02	90.65
			✓	✓	✓	79.10	88.32
		✓	✓	✓	✓	**84.50**	**94.25**

*DA stands for Data Augmentation.

4.2 Analysis

In this section, we show the experiments and results of the contributions proposed in Section 3.2. We show the results of state-of-the-art segmentation models shown in Section 3.2.1 on our own fire scene parsing dataset proposed in Section 3.2.1.1.

4.2.1 Semantic Segmentation for Emergency Analysis

Experimental Setup: We test the performance of the above mentioned segmentation models on our fire emergency scene parsing dataset. For a fair comparison, we split our dataset into training and testing sets which is kept the same for all the models. Out of the 2065 images, we use 1665 images for training and 400 images for testing. We train all the models for 250 epochs on our dataset. To effectively compare the performance of the segmentation models, we use the same backbone/encoder network for all models. We set the ImageNet pretrained ResNet-50 [31] as the backbone network. Since the encoder/backbone is the same for all segmentation models, the fire image classification accuracy is also relatively the same, i.e. around 94.5%. However, to evaluate the segmentation models independently of the encoder/backbone fire detection accuracy, we run the whole model on all test images. The main purpose of multitask learning is to optimize computations during inference.
We have tested the models while conditionally running the decoder conditioned on the backbone classifier, which produced similar final results since all models have the same ResNet-50 backbone network that achieves high fire detection accuracy (94.5%).
Also, to better understand the segmentation results of these models on our unique dataset, we also calculate the frequency weighted mIOU of each model. The frequency weighted mIOU weighs the IOU score by taking into account the number of data points in each class [160]. It can be calculated as:

$$(\sum_{i=1}^{k} t_i)^{-1} \sum_{i=1}^{k} \frac{t_i.n_{ii}}{t_i - n_{ii} + \sum_{j=1}^{k} n_{ij}} \in [0,1] \tag{4.1}$$

where, t_i are the number of pixels belonging to class i in the ground truth segmentation mask. n_{ii} are the number of pixels belonging to class i and are predicted as class i and n_{ij} are the number of pixels belonging to class i and are predicted as class j. The dataset consists of images of different sizes in the RGB format, so we resize all images to 473×473, which are fed as the input to the models. We train the models using the Adam optimizer [54]. The batch-size is set to 16 images per

batch. We use standard data augmentation techniques to improve generalization performance on the test set, such as random scaling, gaussian blur, random rotation and horizontal flipping. The scaling factor ranges from $[0.75, 1.0, 1.25, 1.5]$ with the randomness of scaling, rotation and flipping set to 0.5.
We implement the segmentation models using Keras [155] and Tensorflow [161] with the help of the Keras segmentation library [162]. The images are pre-processed using OpenCV and the image augmentations are implemented using the imgaug library [163].

Results: The performance of the segmentation models on our dataset is shown in Table 1. SegNet, FCN-8, PSPNet, DeepLabv3 and DeepLabv3+ achieve high mIOU scores, with the DeepLabv3+ scoring the highest, nearing 88% mIOU. The U-Net scored a moderate mIOU of 74.8%. However, the U-Net achieves a competitive frequency weighted mIOU of 82.2%. Table 1 shows that all the segmentation models achieve expected performance results according to their results on the benchmark datasets of Cityscapes [164], MIT ADE20K [165] and Pascal VOC 2012 [166]. Table 4.10 shows that all the segmentation models achieve expected performance results according to their results on the benchmark datasets of Cityscapes [164], MIT ADE20K [165] and Pascal VOC 2012 [166].

Table 4.10: Performance Evaluation of Segmentation Models

Model	Frequency weighted mIOU	mIOU
U-Net	0.822	0.748
SegNet	0.876	0.852
FCN-8	0.891	0.862
PSPNet	0.899	0.871
DeepLabv3	0.909	0.879
DeepLabv3+	0.913	0.884

We also test the effect of transfer learning on the task of emergency analysis with fire emergency scene parsing dataset. We use pretrained versions of the PSPNet with trained weights of the Cityscapes, MIT ADE20K and Pascal VOC 2012 datasets. Then, we fine-tune these pretrained models on our fire emergency scene parsing dataset. We use the same experimental setup for these pretrained models. We test different pretrained models to compare their performance and find out whether pre-

training on some benchmark datasets yields better performance than others.
Our findings are shown in Table 4.11. All the pretrained models give relatively the same performance in terms of mIOU and frequency weighted mIOU scores, with some minute differences. The PSPNet pretrained on the Cityscapes dataset give slightly better performance compared to the other two pretrained models. However, the performance comparison of these pretrained models is highly dependent on the experimental setup and training methodology. So, we believe that the results shown in Table 4.11 should be taken with a grain of salt.

Table 4.11: Performance Evaluation of Pre-trained Segmentation Models

Model	**Pretrained on**	**Frequency weighted mIOU**	**mIOU**
PSPNet	MIT ADE20K	0.918	0.901
PSPNet	CityScapes	0.921	0.903
PSPNet	PASCAL VOC 2012	0.912	0.892

We plot the class-wise IOU distribution shown in Figure 4.1. As we can see from Figure 4.1, the pretrained models score slightly higher than other models in most of the classes. But, the major difference in performance can be seen in the 'Stone', 'Wood', 'Plastic' and 'Cloth' classes. Especially in the 'Stone' class, the U-Net performs poorly, since the 'Stone' class has the lowest number of instances in our dataset (that's why U-Net scores higher in frequency weighted mIOU and low on the mIOU scores).
Other than that, we can see the same trend in all class IOU scores: Pretrained models $>$ Other models and DeepLabv3+ $>$ DeepLabv3 $>$ PSPNet $>$ FCN $>$ SegNet $>$ U-Net, with the exception of the 'Stone' class, where the SegNet scores higher. This could be because SegNet can deal with class imbalance problem better than other models. Overall, the models score less on the classes 'Plants', 'Stone' and 'Other'. This is because the classes 'Stone' and 'Plants' have the least number of instances. However, the 'Others' class has one of the higher number of class instances, but since it contains many different types of objects (where the build material cannot be determined with high confidence), it becomes difficult to classify accurately. Some resulting output images of the segmentation models from our fire emergency scene parsing dataset are displayed in Paper D.

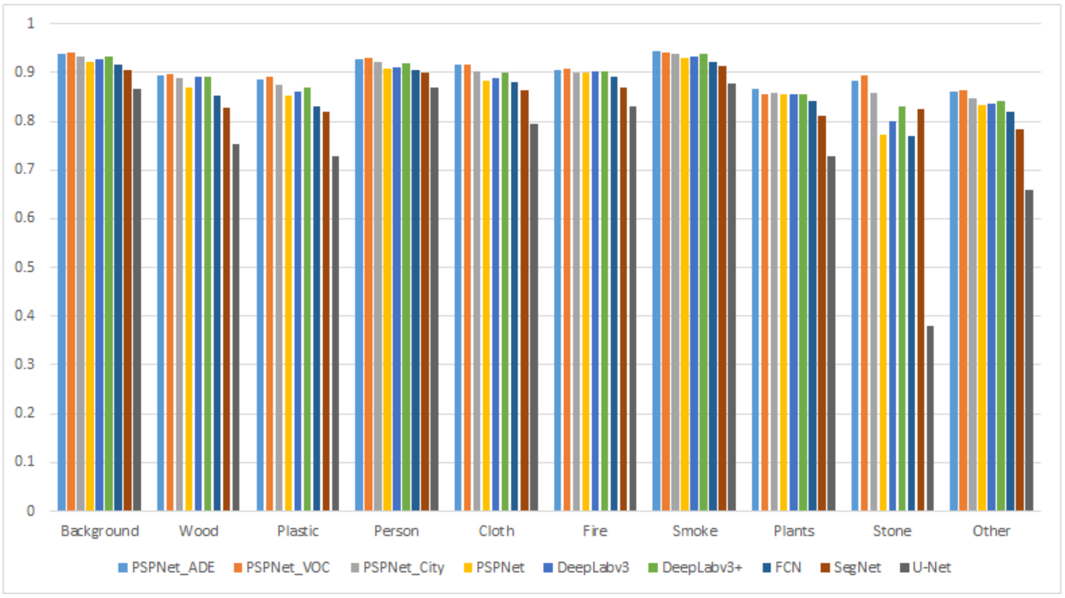


Figure 4.1: Class-wise IOU Distribution

4.3 Evacuation

In this section, we show the results of the reinforcement learning agent proposed in Section 3.3 on the our fire evacuation environment from Section 3.3.2.

4.3.1 Q-matrix Pretrained Deep Q-Networks on the Fire Evacuation Environment

Experimental Setup: We perform unbiased experiments on the fire evacuation environment and compare our proposed approach with state-of-the-art reinforcement learning algorithms. We test different configurations of hyperparameters and show the results with best performing hyperparameters for these algorithms on our environment. The main intuition behind using Q-learning pretrained DQN model was to provide it with important information before hand, to increase stability and convergence. The results confirms our intuition empirically.

The Agent's Network: Unlike the convolutional neural networks [4] used in DQN [49, 50], DDQN [56, 57] and Dueling DQN [58], we implement a fully connected feedforward neural network. The network consists of 5 layers. The ReLU function [167] is used for all layers, except the output layer, where a linear activation is used to produce the output.

Environment: The state of the environment is given as : $S = [10, 10, 10, 10, 0]$ with bottleneck $\mathcal{B} = 10$. All rooms contain 10 people (the exit is empty), which is the maximum possible number of people. We do this to test the agents under maximum stress. The fire starts in room 2 and the fire spread is more towards room 1 than room 3. Room 4 is the exit. The total number of actions possible for this environment is 25. So, the agent has to pick one out of 25 actions at each step.

Training: The Adam optimizer [54] with default parameters and a learning rate η of 0.001 is used for training for all the agents. Each agent is trained for 500 episodes. Training was performed on a 4GB NVIDIA GTX 1050Ti GPU. The models were developed in Python with the help of Tensorflow [168] and Keras [155].

Implementation: Initially, the graph connections were represented as 2D arrays of the adjacency matrix M_A. But, when the building model's graphs get bigger, the adjacency matrices become more and more sparse, which makes the 2D array representation inefficient. So, the most efficient and easiest way to implement a graph is as a dictionary, where the keys represent rooms and their values are an array that lists all the rooms that are connected to it.

$$dict_{graph} = \{room_i : [room_j; \quad \forall j \quad in \quad M_{A_{i,j}} = 1]\}$$

Results: We first compare the Q-matrix pretrained versions of the DQN and its variants with the original models. The graph based comparisons between models consists of number of time-steps for evacuating all people on the y-axis and episode number on the x-axis. We put an upper-limit of 1000 time-steps for an episode due to computational reasons. The training loop breaks and a new episode begins once this limit is reached.

The graph comparing DQN with our proposed Q-matrix pretrained DQN (QMP-DQN) in Figure 4.2 shows the difference in their performance on the fire evacuation environment. Although the DQN reaches the optimal number of time-steps quickly, it isn't able to stay there. The DQN drastically diverges from the solution and is highly unstable.

It's the same case with DDQN (Figure 4.3) and Dueling DQN (Figure 4.4), which, although perform better that DQN with less fluctuations and spend more time near the optimal solution. Our results clearly shows a big performance lag compared to the pretrained versions. As these results suggest that pretraining ensures convergence and stability. We show that having some important information about the environment prior to training reduces the complexity of the learning task for an agent.

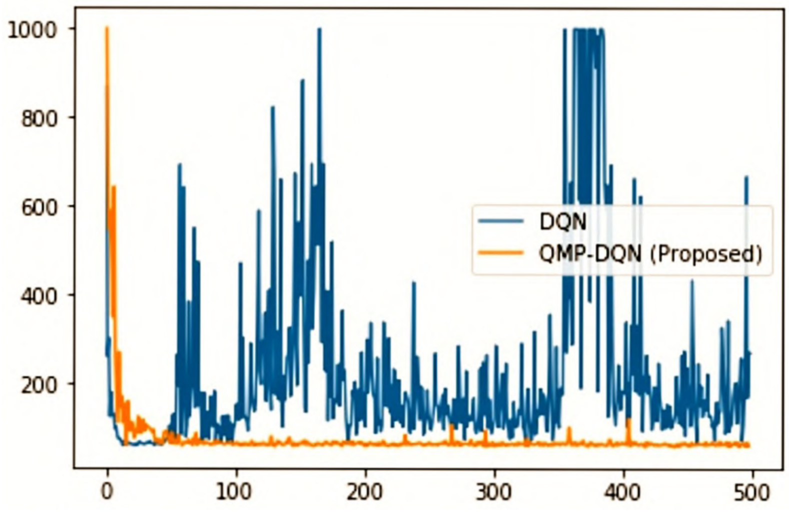


Figure 4.2: Q-matrix pretrained DQN vs DQN

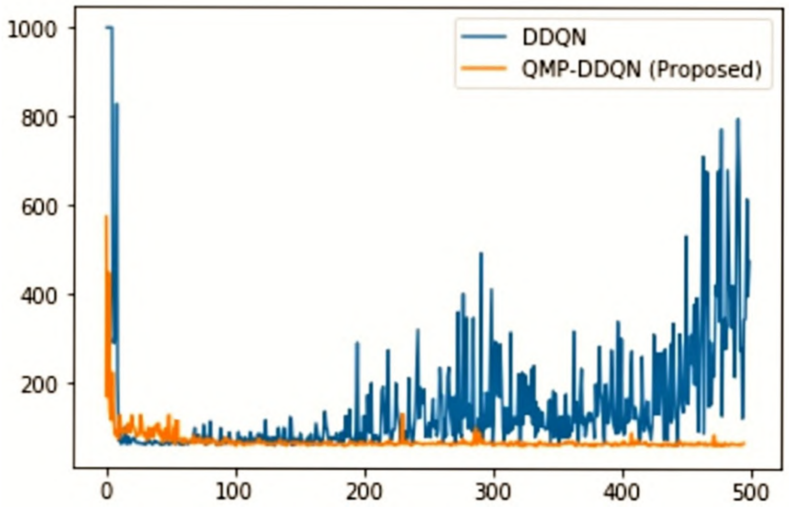


Figure 4.3: Q-matrix pretrained DDQN vs DDQN

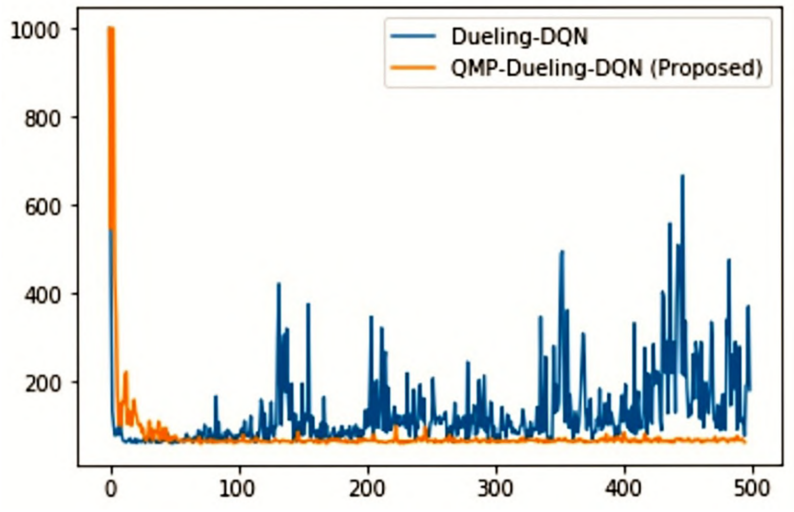


Figure 4.4: Q-matrix pretrained Dueling DQN vs Dueling DQN

Table 4.12: Performance

Model	**Avg. Steps**	**Avg. Steps with PT**	**Min. Steps**	**Min. Steps with PT**	**Training Time**	**Training time with PT**
DQN	228.2	**76.356**	63	**61**	10.117	**6.87**
DDQN	134.62	**71.118**	61	**60**	12.437	**8.11**
Dueling DQN	127.572	**68.754**	61	**60**	12.956	**9.02**

The original Q-learning based models aren't able to cope with the dynamic and stochastic behaviour of the environment. And since they don't posses pretrained information, their learning process is made even more difficult. Table 4.12 displays a few numerical results, comparing DQN, DDQN and Dueling DQN, with and without the Q-matrix pretraining on the basis of average number of time-steps for all 500 episodes, minimum number of time-steps reached during training and the training time per episode.

As it was also clear from the Figures 4.2, 4.3 and 4.4, the average number of time-steps is greatly reduced with pretraining, as it makes the models more stable by reducing variance. All the DQN based models are able to come close to this, but pretraining pushes these models further and achieves the minimum possible number of time-steps. Even though the difference seems small, in emergency situations even the smallest differences could mean a lot at the end. The training time is also reduced with pretraining, as the number of time-steps taken during training is reduced and pretrained models get a better starting position nearer to the optimum.

Next, we make comparisons between our proposed approach and state-of-the-art

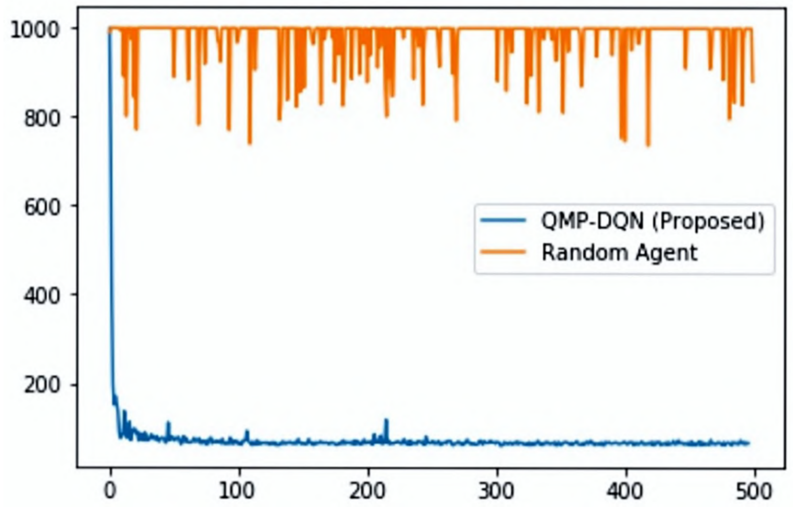


Figure 4.5: Proposed method vs Random Agent

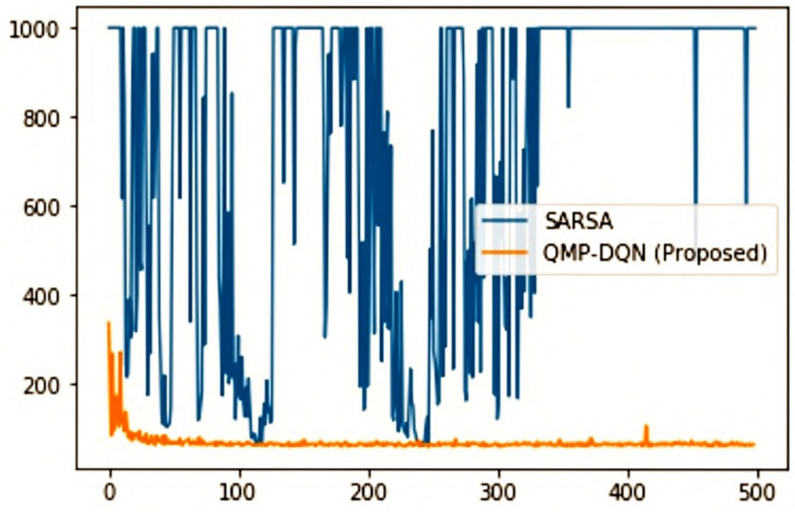


Figure 4.6: Proposed method vs State-Action-Reward-State-Action method

reinforcement learning algorithms. For these comparisons, we use the Q-matrix pretrained Dueling DQN model, abbreviated QMP-DQN. We also compare it with a random agent, shown in Figure 4.5. The random agent performs random actions at each step, without any exploration. The random agent's poor performance of 956.33 average time-steps shows that finding the optimal or even evacuating all the people isn't a simple task.

The State-Action-Reward-State-Action (SARSA) algorithm is an on-policy reinforcement learning agent introduced in [141]. While Q-learning follows a greedy policy, SARSA takes the policy into account and incorporates it into its updates. It updates values by considering the policy's previous actions. On-policy methods like SARSA have a downside of getting trapped in local minima if a sub-optimal

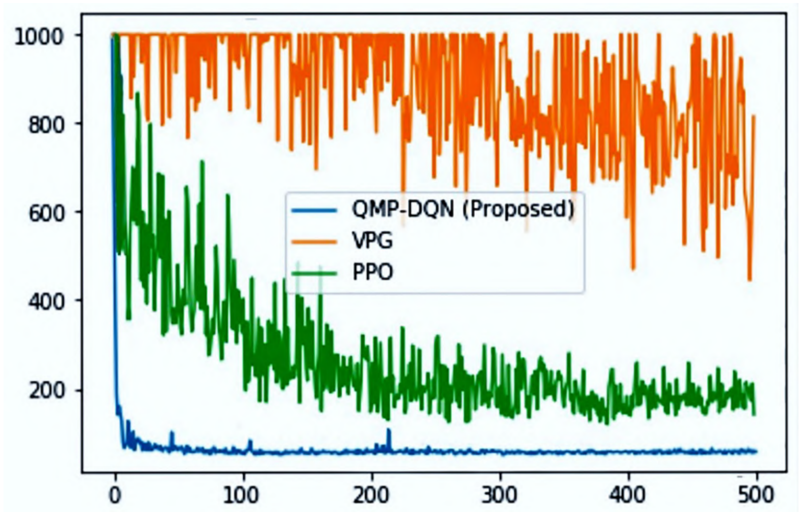


Figure 4.7: Proposed method vs Policy based methods (PPO and VPG)

policy is judged as the best. On the other hand, off-policy methods like Q-learning are flexible and simple as they follow a greedy approach. As it is clear from Figure 4.6, that SARSA behaves in a highly unstable manner and isn't able to reach the optimal solution and shows high variance.

Policy gradient methods are highly preferred in many applications, however they aren't able to perform optimally on our fire evacuation environment. Since the optimal policy could change in a few time-steps in our dynamic environment, greedy action selection is probably the best approach. An evacuation path that seems best at a particular time step could be extremely dangerous after the next few time-steps and a strict policy of routing cannot be followed continuously due to fire spread and/or bottleneck. These facts are evident from Figure 4.7, where we compare our approach to policy gradient methods like Proximal Policy Optimization (PPO) [169] and Vanilla Policy Gradient (VPG) [170]. Even though PPO shows promising movement, it isn't able to reach the optimum.

Another major type of reinforcement learning algorithms are the actor-critic methods. It is a hybrid approach consisting of two neural networks: an actor which controls the policy (policy based) and a critic which estimates action values (value based). To further stabilize the model, an advantage function is introduced which gives the improvement of an action compared to an average action used in a particular state. Apart from the previously mentioned shortcomings of using policy based methods on the fire evacuation environment, the advantage function would have high variance since the best action at a particular state could change rapidly leading to unstable performance. This is apparent from Figure 4.8, where we compare the synchronous advantage actor critic method (A2C) [171] with our proposed method.

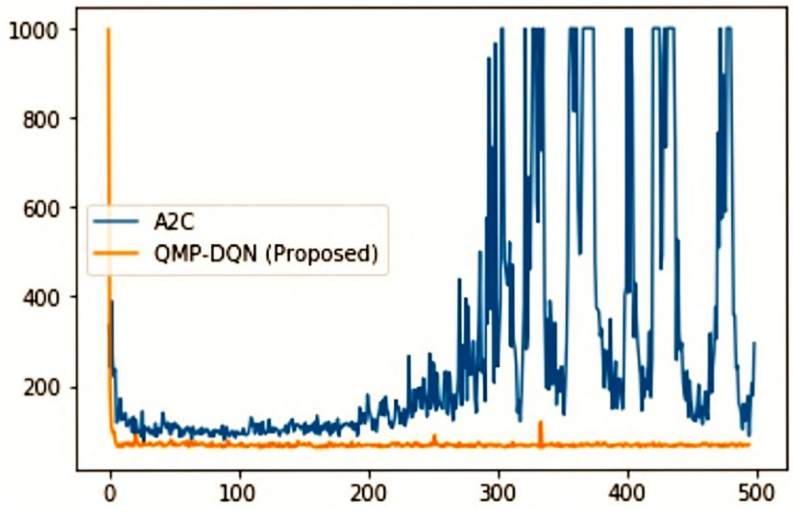


Figure 4.8: Proposed method vs Synchronous Advantage Actor Critic method (A2C)

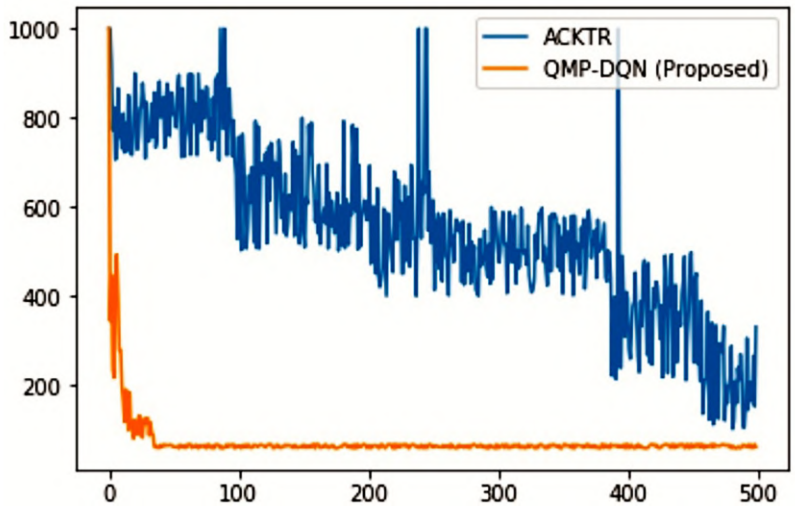


Figure 4.9: Proposed method vs Actor Critic using Kronecker-Factored Trust Region (ACKTR)

Table 4.13: Comparison with State-of-the-art RL Algorithms

Model	**Avg. steps**	**Min. steps**	**Training Time (per episode)**
SARSA	642.21	65	19.709
PPO	343.75	112	16.821
VPG	723.47	434	21.359
A2C	585.92	64	25.174
ACKTR	476.56	79	29.359
Random Agent	956.33	741	-
QMP-DQN (Dueling DQN Backbone)	**68.754**	**60**	**9.02**

The A2C gives near optimal performance in the beginning but diverges and rapidly fluctuates.

We do not compare our proposed method with the asynchronous advantage actor critic method (A3C) [172], because A3C is just an asynchronous version of A2C, which is more complex as it creates many parallel versions of the environment and gives relatively the same performance, but is not as sample efficient as claimed in [173]. The only advantage of A3C is that it exploits parallel and distributed CPU and GPU architectures which boosts learning speed as it can update asynchronously. However, the main focus of this paper is not learning speed. Hence, we think that the comparison with A2C is sufficient for actor-critic models.

Probably the best performing Actor Critic based model is the ACKTR (Actor Critic with Kronecker-factored Trust Region) [174]. The algorithm based on applying trust region optimization using Kronecker-factored approximation, which is the first scalable trust region natural gradient method for actor critic models that can be applied to both continuous and discrete action spaces. The Kronecker-factored Approximate Curvature (K-FAC) [175], is used to approximate the Fisher Matrix to perform approximate natural gradient updates. We compare our method to the ACKTR algorithm, shown in Figure 4.9. The results suggest that the ACKTR is not able to converge (within 500 episodes, due to slow convergence rate) and is susceptible to the dynamic changes in the environment as evident from the fluctuations. ACKTR is far too complex compared to our proposed method, which converges much faster and deals with the dynamic behaviour of the fire evacuation environment efficiently.

We summarize our results in Table 4.13. All the RL agents use the same network configuration mentioned in Table 1 for unbiased comparison. The training time for the QMP-DQN is much lower compared to other algorithms because pretraining provides it with a better starting point, so it requires less number of time-steps and memory updates to reach the terminal state. Also, SARSA and A2C come really close to the minimum number of time-steps, but as the average number of time-steps suggests, they aren't able to converge and exhibit highly unstable performance. Our proposed method, Q-matrix pretrained Dueling Deep Q-network gives the best performance on the fire evacuation environment by a huge margin.

Note that, in all the comparison graphs, our proposed method comes close to the global optimum, but isn't able to completely converge to it. This is because of the uncertainty probability p, which decides whether an action is performed or not and is set to 0.15. This uncertainty probability is used to map the uncertain crowd behaviour. Even though, p, does not allow complete convergence, it also prevents the model from memorizing an optimal path which might change as the fire spreads.

4.3.2 Scalability: Large and Complex Real World Scenario - University of Agder Building

Here, we show the results of using our model with the one-step simulation and action importance attention mechanism, proposed in Section 3.3.4, on the UiA building.

Experimental Setup: The graph for UiA's building model is based on the actual structure of the 2nd floor of blocks A, B, C and D[1]. The graph for the building model can be found in Paper C. It consists of 91 rooms (from room 0 to room 90) out of which there are 10 exits. We simulate the fire evacuation environment in which there are multiple distributed fires in rooms 14, 29, 59 and 80. The fire spread for each fire is individually simulated in a random direction as shown by the yellow nodes in the graph.

The building connectivity can be quite complex and there has been limited research work that deals with this aspect. The graph structure shows that these connections between rooms cannot possibly be captured by a grid based or maze environment. Also, note that, the double sided arrows in the graph enable transitions back and forth between rooms. This makes the environment more complicated for the agent since the agent could just go back and forth between 'safe' rooms and get stuck in a loop and may never converge. This point makes pretraining even more indispens-

[1]UiA building map can be found here: `https://use.mazemap.com/#v=1&zlevel=2&left=8.5746533&right=8.5803711&top=58.3348318&bottom=58.3334208&campusid=225`

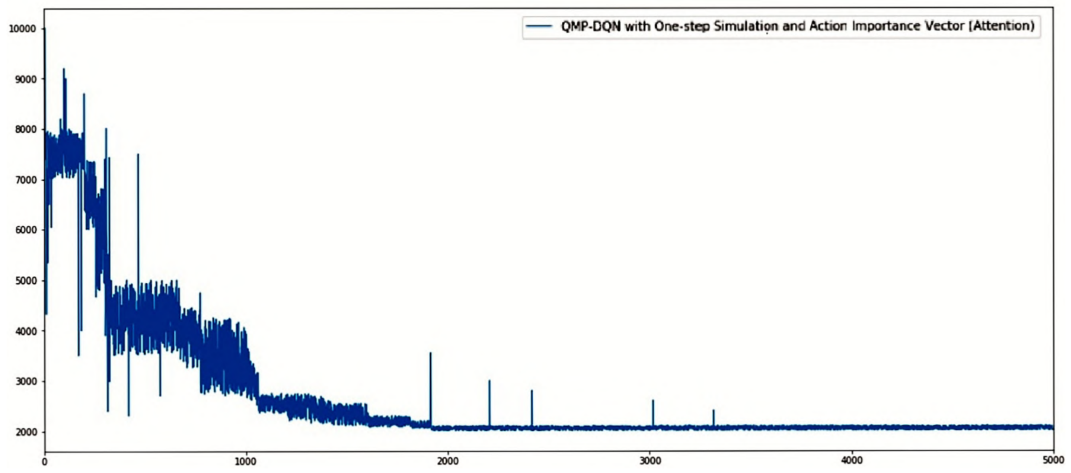


Figure 4.10: Proposed method applied on the UiA Building

able.
Since, the proposed method is able to reduce the action space by a lot, the neural network doesn't need to be made too large. Note that the addition layer does not require any trainable parameters. The neural network is trained using the Adam optimizer [54] with default hyperparameter settings and a learning rate $\eta = 0.001$ for 5000 episodes. The training was performed on the NVIDIA DGX-2. The optimal number of steps for evacuation in the UiA building graph is around ~ 2000.

Results: The results of our proposed method consisting of shortest path Q-matrix transfer learning to Dueling-DQN model with one-step simulation and action importance vector acting as an attention mechanism applied on the University of Agder's A,B,C and D blocks consisting of 91 rooms and 8281 actions (whose graph is shown in Fig. 13) is shown in Fig. 14. The performance numbers are given below:

- **Average number of time-steps:** 2234.5
- **Minimum number of time-steps:** $\sim$2000
- **Training time (per episode):** 32.18 s

The graph in Figure 4.10 shows the convergence of our method with evacuation time-steps on the y-axis and the episode number on the x-axis. It takes slightly longer to converge compared to the convergence in previous small example environments. This is obviously due to the size of the environment and complex connectivity. But overall the performance of our model is excellent.
After $\sim$ 1900 episodes, the algorithm has almost converged. There are a few spikes

suggesting fluctuations from the optimal behaviour due to the dynamic nature of the environment and the uncertainty in actions. After ~ 3300 episodes, the algorithm completely converges in the range $(2000 - 2070)$ times-steps for total evacuation. The method cannot converge to the minimum possible time-steps $= 2000$ because of the fire spread dynamics, encountering bottleneck conditions and action uncertainty.
The results clearly suggest that even though the proposed fire evacuation environment is dynamic, uncertain and full of constraints, our proposed method using novel action reduction technique with attention based mechanism and transfer learning of shortest path information is able to achieve excellent performance on a large and complex real world building model. This further confirms that, with a minute added overhead of one-step simulation and action importance vector, our method is scalable to much larger and complex building models.

4.4 Summary

In summary, this chapter shows the results of our proposed models for detection, analysis and evacuation phases. We show that for visual detection, transfer learning with state-of-the-art deep CNN models gives good performance which is further boosted by using ELM as the classifier. In case of audio detection, our proposed attention based deep CNN model with multiple feature channels sets a new state-of-the-art on three benchmark ESC datasets. We use a reduced version of this model to perform sound based emergency detection that achieves near optimal performance. For the analysis stage, our proposed way of analysis using semantic segmentation with state-of-the-art models achieves excellent results, especially with models pretrained on benchmark segmentation datasets. Finally, for the evacuation step, the proposed method of Q-matrix pretrained DQN agent outperforms other state-of-the-art reinforcement learning techniques by a large margin. Also, our proposed one-step simulation and action importance attention mechanism combined with the Q-matrix pretrained DQN is able to achieve near optimal performance on the large and complex real world UiA evacuation problem. Hence, we are able to experimentally show that our proposed models for the fire emergency management stages are either able to outperform currently used methods or set a new paradigm of methods that have never been used before.

Chapter 5

Conclusion and Future Work

In this thesis, we propose novel deep learning and neural network methods and architectures to solve the shortcomings of stages of fire emergency management. We divide the problem of fire emergencies into three stages: Detection, Analysis and Evacuation. Detection stage deals with detecting whether there is an (fire) emergency or not. The emergency situation is analysed to retrieve relevant information in the analysis stage. Finally, in the evacuation stage, the aim is to evacuate maximum number of people in the minimum number of time steps.

5.1 Conclusion to Research Questions

In this section, we conclude the findings of our proposed methods in accordance to the research questions in Chapter 1.

Research Question 1: We validate using state-of-the-art CNNs on our fire detection dataset. Our experiments show that a shallow CNN overfits on the majority class, while VGG16 and ResNet50 with transfer learning do not overfit. We also experiment with the number of fully connected layers in the VGG16 and ResNet50 models with pretrained ImageNet weights. They are able to achieve 91.18% and 92.15% accuracy respectively.

Research Question 2: We compare our proposed CNN-ELM hybrid model for fire detection, with the standard CNN and CNN-SVM hybrid models. The results show that the VGG16-ELM and ResNet50-ELM models require less training data and much lower training time. Also, there is an increase in accuracy of about 2.8% to 7.1% across the models and an inference time speed-up of 5x to 6x.

Research Question 3: The experiments conducted on the fire evacuation environment show that pretraining the DQN agent is highly advantageous. Our pretrained DQN based agents are able to outperform state-of-the-art RL techniques such as DQN, DDQN, Dueling DQN, PPO, VPG, A2C, ACKTR, SARSA etc. in terms of time-steps required to evacuate people. Using the action importance attention mechanism, we are able to scale the model to work on a real world large scale evacuation scenario based on the UiA building by reducing the action space from 8281 to 819 actions.

Research Question 4: We use state-of-the-art semantic segmentation models like U-Net, SegNet, FCN, PSPNet, DeepLabv3 and DeepLabv3+ to segment objects based on their build material in the emergency environment. The fire scene parsing dataset consists of 10 object classes. We also test the effects of transfer learning from different sets of pretraining weights. Our results show that, pretrained models from different benchmark datasets have relatively the same effect. We also compare models based on the frequency weighted mIOU scores to balance the effects of imbalance in classification. The U-Net, SegNet, FCN, PSPNet, DeepLabv3 and DeepLabv3+ achieve 0.822, 0.876, 0.891, 0.899, 0.909 and 0.913 frequency weighted mIOU respectively. Transfer learning is able to further boost the performance of the PSPNet to 0.921 frequency weighted mIOU, pretrained on the Cityscapes dataset.

Research Question 5: The proposed novel ESC model is compared with previous state-of-the-art environment sound classification models. We also show results for different feature combinations and ablations to the CNN architecture to justify our design choices. By achieving 94.75%, 87.45% and 97.52% accuracy on the ESC-10, ESC-50 and UrbanSound8K benchmark datasets respectively, our model sets a new state-of-the-art performance for environment sound classification.

Research Question 6: Our proposed model is tested on a combination of the benchmark ESC and UrbanSound8K datasets. The sound classes are arranged into two categories: emergency and non-emergency sounds. The emergency sounds consist of sirens, fire crackling, glass breaking, gun shot etc. and the rest fall in the non-emergency category. Since the dataset is highly imbalanced, we use class weights while calculating the cost function. The model is able to achieve a 99.56% emergency detection rate.

In the next sections, we elaborate on our concluding remarks for the proposed models for detection, analysis and evacuation and discuss some future research work that might improve our methods and the overall system.

5.2 Detection

5.2.1 Visual Detection

For visual detection, we have proposed to use two state-of-the-art Deep Convolutional Neural Networks for fire detection in images, VGG16 and Resnet50. We test these models on our dataset which is made specifically to replicate real world environment. The dataset includes images that are difficult to classify and is highly unbalanced by including less fire images and more non-fire images since fire is a rare occurrence in the real world. We rationalize the use of such deep and complex models by showing that a simple CNN performs poorly on our dataset. To further increase accuracy, we added fully connected layers to both VGG16 and Resnet50. Results show that adding fully connected layers does improve the accuracy of the detector but also increases its training time. In practice, increasing the number of fully connected layers by more than one results in minute increase in accuracy compared to the large increase in training time, even if the models are pre-trained. We found that deep CNNs provide good performance on a diverse and highly imbalanced dataset of small size, with Resnet50 slightly outperforming VGG16 and adding fully connected layers slightly improves accuracy but takes longer to train.
To further improve visual detection, training time and inference speed, we have proposed a hybrid model for fire detection. The hybrid model combines the feature extraction capabilities of the Deep CNNs mentioned above and the classification ability of ELM. The fully connected layers are removed completely and replaced by a single hidden layer feedforward neural network trained using the ELM algorithm. This decreases complexity of the network and increases speed of convergence. We test our model on our fire dataset. Our hybrid model is compared with the original VGG16 and Resnet50 models and also with SVM hybrid models. Our Deep CNN-ELM model is able to outperform all other models in terms of accuracy by 2.8% to 7.1% and training time by a speed up of 20x to 51x and requires less training data to achieve higher accuracy for the problem of fire detection.

5.2.2 Audio Detection

In order to build a model for emergency detection using sounds, we develop a novel model for the environment sound classification task. It consists of multiple feature channels and attention based deep convolutional neural network with domain wise

convolutions. We combine feature extraction methods like the MFCC, GFCC, CQT and Chromagram to create a multi channel input for the CNN classifier. The model consists of two block: Main block and Attention block. We employ a Deep CNN consisting of separable convolutions in the main block. The separable convolutions work on the time and feature domains separately. Parallel to the main blocks, we also use an attention mechanism that consists of depthwise separable convolution. Both channel and spatial attention are achieved using a small increase in number of parameters. We test our model on the three benchmark datasets: ESC-10, ESC-50 and UrbanSound8K. We use mix-up data augmentation techniques to further improve performance. Our model achieves 94.75%, 87.45% and 97.52% accuracy on ESC-10, ESC-50 and UrbanSound8K respectively, which sets the new state-of-the-art on all three datasets.
We use a reduced version of this model for emergency detection using audio signals. We reduce the size of the model by removing the attention block and reducing the number of feature channels in the model. We test our model by combining the ESC-50 and UrbanSound8K benchmark datasets and categorizing emergency sound related classes in one category and the rest in another category to make a binary classification problem. We also use class weights since the data is quite imbalanced. Our model achieves 99.56% accuracy on the combination of the ESC-50 and UrbanSound8K datasets.

5.3 Analysis

For the analysis step, we have proposed a new approach towards fire emergency analysis using image segmentation. To analyse a fire emergency situation, we propose to identify and classify objects based on their build material in order to display inflammable and non-inflammable objects. For this purpose, we built our own fire scene parsing dataset consisting of 10 object classes in 2065 images. To segment images based on object build materials, we employed state-of-the-art segmentation models: U-Net, SegNet, FCN, PSPNet, DeepLabv3 and DeepLabv3+. Comparison between these models shows that SegNet, FCN-8, PSPNet, DeepLabv3 and DeepLabv3+ give good performance, with DeepLabv3+ scoring slightly more than others. To reduce computation during inference, we use multitask learning to use th encoder/backbone as a preliminary classifier. If it detects fire in an image, only then the decoder is activated to segment the image. We also showed the importance of transfer learning by fine-tuning pretrained PSPNet models on our dataset. We also compared pretrained models based on the benchmark dataset that they have been trained on. Results show that the PSPNet pretrained on the Cityscapes dataset gives slightly better performance compared to the model trained on MIT ADE20K and Pascal VOC 2012. However, we would like to point out that since there is a very small difference in performance, the results might differ for different training sets and schemes.

5.4 Evacuation

For the final stage of evacuation, we propose a novel reinforcement learning approach. We build the first realistic fire evacuation environment to train reinforcement learning agents. The environment is implemented in OpenAI gym format. The environment has been developed to simulate realistic fire scenarios. It includes features like fire spread with the help of exponential decay reward functions and degree functions, bottlenecks, uncertainty in performing an action and a graph based environment for accurately mapping a building model.
We also propose a new reinforcement learning method for training on our environment. We use tabular Q-learning to generate q-values for shortest path to the exit using the adjacency matrix of the graph based environment. Then, the result of Q-learning (after being offset by a σ) is used to pretrain the DQN network weights to incorporate shortest path information in the agent. Finally, the pretrained weights of the DQN based agents are trained on the fire evacuation environment.
We prove the faster convergence of our method using Task Transfer Q-learning theorems and the convergence of Q-learning for the shortest path task. The Q-matrix pretrained DQN agents (QMP-DQN) are compared with state-of-the-art reinforcement learning algorithms like DQN, DDQN, Dueling-DQN, PPO, VPG, A2C, ACKTR and SARSA on the fire evacuation environment. The proposed method is able to outperform all these models on our environment on the basis of convergence, training time and stability. Also, the comparisons of QMP-DQN with original DQN based models show clear improvements over the latter.
Finally, we show the scalability of our method by testing it on a real world large and complex building model. In order to reduce the large action space (8281 actions), we use the one-step simulation technique on the pretraining environment instance to calculate the action importance vector, which can be thought of as an attention based mechanism. The action importance vector gives the best k actions a weight of 0 and the rest are assigned a large negative weight of -9999 (to render the Q-values of these too low to be selected by the Q-function). This reduces the action space by $\sim 90\%$ and our proposed method, QMP-DQN model, is applied on this

reduced action space. We test this method on the UiA, Campus Grimstad building, with the environment consisting of 91 rooms. The results show that this combination of methods works really well in a large real world fire evacuation emergency environment.

5.5 Future Work

The models proposed in this thesis pertain to each fire emergency stage separately. They have been designed in a disjoint manner to support each phase of the process as a stand alone model. However, to truly create an AI based fire emergency system, we would like to combine all these separate models into one large neural network that is able to assist through the whole process without having to transition between different models at different steps.

We would like to work on creating the first ever end-to-end trainable deep neural network model with combined computer vision and reinforcement learning capabilities. There are a lot of issues that need to be addressed in building such a model, especially in terms of data. Also, using different characteristic loss functions for each stage. However, deep supervision and auxiliary losses can be used to deal with this obstacle.

Another dilemma that we might encounter during integration of these phases is the information flow between different application modules. Since the outputs of each stage are non-gradient variables, they cannot be trained end-to-end by simply combining them since gradient cannot flow through these output variables. But, we can use the reparametrization trick introduced in [176] that might prove helpful in this case. Also, a thorough individual stage analysis and performance improvement is certainly possible that we would like to carry out in the future.

We would also like to test the whole fire emergency management system on a real emergency case. And, integrate the system into the existing fire emergency department systems. In order to do this, each phase must be able to work in real-time. So, another future task would be to efficiently implement and/or modify these models in hardware devices (Raspberry Pi, Camera systems) to reduce inference time.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[2] S. R. Institute, “Natural catastrophes and man-made disasters in 2017: a year of record-breaking losses,” *Sigma Report*, vol. 1, p. 59, 2018.

[3] W. Sun, P. Bocchini, and B. D. Davison, “Applications of artificial intelligence for disaster management,” *Natural Hazards*, pp. 1–59, 2020.

[4] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[5] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[6] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudik, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[7] A. Krizhevsky, “Convolutional deep belief networks on cifar-10,” 2010.

[8] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, August 1980.

[9] M. Sahidullah and G. Saha, “Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition,” *Speech Communication*, vol. 54, no. 4, pp. 543 – 565, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639311001622

[10] M. Müller, *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer-Verlag, 2007.

[11] V. Tyagi and C. Wellekens, "On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 1, March 2005, pp. I/529–I/532 Vol. 1.

[12] B. Logan, "Mel frequency cepstral coefficients for music modeling." in *ISMIR*, vol. 270, October 2000, pp. 1–11.

[13] Y. Shao, Z. Jin, D. Wang, and S. Srinivasan, "An auditory-based feature for robust speech recognition," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 4625–4628.

[14] S. Chachada and C. . J. Kuo, "Environmental sound recognition: A survey," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, Oct 2013, pp. 1–9.

[15] W. Burgos, "Gammatone and mfcc features in speaker recognition (doctoral dissertation)," Ph.D. dissertation, 2014.

[16] C. Schörkhuber, "Constant-q transform toolbox for music processing," 2010.

[17] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[18] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet(cyclegan(cqt(audio))) pipeline for musical timbre transfer," *CoRR*, vol. abs/1811.09620, 2018. [Online]. Available: http://arxiv.org/abs/1811.09620

[19] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," *CoRR*, vol. abs/1706.07156, 2017. [Online]. Available: http://arxiv.org/abs/1706.07156

[20] T. Lidy and A. Schindler, "Cqt-based convolutional neural networks for audio scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, vol. 90. DCASE2016 Challenge, 2016, pp. 1032–1048.

[21] R. N. Shepard, "Circularity in judgments of relative pitch," *The Journal of the Acoustical Society of America*, vol. 36, no. 12, pp. 2346–2353, 1964.

[22] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis." in *International Society for Music Information Retrieval*, Aug 2010, pp. 625–636.

[23] D. Ellis, "Chroma feature analysis and synthesis," April 2007. [Online]. Available: https://labrosa.ee.columbia.edu/matlab/chroma-ansyn/

[24] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.

[25] ——, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[26] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[27] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[28] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning Proceedings 1995*. San Francisco (CA): Morgan Kaufmann, 1995, pp. 30 – 37. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978155860377650013X

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016. [Online]. Available: http://arxiv.org/abs/1610.02357

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," pp. 1097–1105, 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[35] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4960–4964.

[36] C.-C. Chiu, T. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," 2018. [Online]. Available: https://arxiv.org/pdf/1712.01769.pdf

[37] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, T. Jebara and E. P. Xing, Eds. JMLR Workshop and Conference Proceedings, 2014, pp. 1764–1772. [Online]. Available: http://jmlr.org/proceedings/papers/v32/graves14.pdf

[38] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[39] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *CoRR*, vol. abs/1303.5778, 2013. [Online]. Available: http://arxiv.org/abs/1303.5778

[40] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 8614–8618.

[41] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, Jan 2012.

[42] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 649–657. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969312

[43] R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," *CoRR*, vol. abs/1804.03235, 2018. [Online]. Available: http://arxiv.org/abs/1804.03235

[44] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, N. Parmar, M. Schuster, Z. Chen, Y. Wu, and M. Hughes, "The best of both worlds: Combining recent advances in neural machine translation," *CoRR*, vol. abs/1804.09849, 2018. [Online]. Available: http://arxiv.org/abs/1804.09849

[45] L. Sestorain, M. Ciaramita, C. Buck, and T. Hofmann, "Zero-shot dual machine translation," *CoRR*, vol. abs/1805.10338, 2018. [Online]. Available: http://arxiv.org/abs/1805.10338

[46] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, and M. Zhou, "Achieving human parity on automatic chinese to english news translation," *CoRR*, vol. abs/1803.05567, 2018. [Online]. Available: http://arxiv.org/abs/1803.05567

[47] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–8.

[48] H. D. Patino and D. Liu, "Neural network-based model reference adaptive control system," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 30, no. 1, pp. 198–204, Feb 2000.

[49] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[50] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, feb 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[51] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 09 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729586

[52] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[53] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-24, Mar 2010. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[55] L.-J. Lin, "Reinforcement learning for robots using neural networks," Ph.D. dissertation, Pittsburgh, PA, USA, 1992, uMI Order No. GAX93-22750.

[56] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2613–2621. [Online]. Available: http://papers.nips.cc/paper/3964-double-q-learning.pdf

[57] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on*

Artificial Intelligence, ser. AAAI'16. AAAI Press, 2016, pp. 2094–2100. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016191

[58] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 1995–2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045601

[59] S. Verstockt, P. Lambert, R. Van de Walle, B. Merci, and B. Sette, "State of the art in vision-based fire and smoke dectection," in *International Conference on Automatic Fire Detection, 14th, Proceedings*, H. Luck and I. Willms, Eds., vol. 2. University of Duisburg-Essen. Department of Communication Systems, 2009, pp. 285–292.

[60] J. Shao, G. Wang, and W. Guo, "An image-based fire detection method using color analysis," in *2012 International Conference on Computer Science and Information Processing (CSIP)*, Aug 2012, pp. 1008–1011.

[61] D. Y. T. Chino, L. P. S. Avalhais, J. F. R. Jr., and A. J. M. Traina, "Bowfire: Detection of fire in still images by integrating pixel color and texture analysis," *CoRR*, vol. abs/1506.03495, 2015. [Online]. Available: http://arxiv.org/abs/1506.03495

[62] B. U. Töreyin, Y. Dedeoglu, U. Güdükbay, and A. E. Çetin, "Computer vision based method for real-time fire and flame detection," *Pattern Recognition Letters*, vol. 27, no. 1, pp. 49 – 58, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865505001819

[63] K. Poobalan and S. Liew, "Fire detection algorithm using image processing techniques," in *3rd International Conference on Artificial Intelligence and Computer Science (AICS2015)*, Ocotober 2015.

[64] J. Vicente and P. Guillemant, "An image processing technique for automatically detecting forest fire," *International Journal of Thermal Sciences*, vol. 41, no. 12, pp. 1113 – 1120, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1290072902013972

[65] T. Toulouse, L. Rossi, T. Celik, and M. Akhloufi, "Automatic fire pixel detection using image processing: a comparative analysis of rule-based and ma-

chine learning-based methods," *Signal, Image and Video Processing*, vol. 10, no. 4, pp. 647–654, 2016.

[66] S. Frizzi, R. Kaabi, M. Bouchouicha, J. M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 877–882.

[67] Q. Zhang, J. Xu, L. Xu, and H. Guo, "Deep convolutional neural networks for forest fire detection," February 2016.

[68] C. Tao, J. Zhang, and P. Wang, "Smoke detection based on deep convolutional neural networks," in *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, Dec 2016, pp. 150–153.

[69] B. T. Polednik, "Detection of fire in images and video using cnn," *Excel@FIT*, 2015.

[70] G. Bradski, "Opencv," *Dr. Dobb's Journal of Software Tools*, 2000.

[71] J. Z. Z. Z. C. Q. Y. K. D. Zhang, S. Han and X. Chen, "Image based forest fire detection using dynamic characteristics with artificial neural networks," in *2009 International Joint Conference on Artificial Intelligence*, April 2009, pp. 290–293.

[72] W.-B. Horng and J.-W. Peng, "Image-based fire detection using neural networks," in *JCIS*, 2006.

[73] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014. [Online]. Available: http://arxiv.org/abs/1403.6382

[74] J. Nagi, G. A. D. Caro, A. Giusti, F. Nagi, and L. M. Gambardella, "Convolutional neural support vector machines: Hybrid visual pattern classifiers for multi-robot systems." in *ICMLA (1)*. IEEE, 2012, pp. 27–32. [Online]. Available: http://dblp.uni-trier.de/db/conf/icmla/icmla2012-1.html#NagiCGNG12

[75] J. v. d. Wolfshaar, M. F. Karaaba, and M. A. Wiering, "Deep convolutional neural networks and support vector machines for gender recognition," in *2015 IEEE Symposium Series on Computational Intelligence*, Dec 2015, pp. 188–195.

[76] Y. Tang, "Deep learning using support vector machines," *CoRR*, vol. abs/1306.0239, 2013. [Online]. Available: http://arxiv.org/abs/1306.0239

[77] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," *CoRR*, vol. abs/1406.5774, 2014. [Online]. Available: http://arxiv.org/abs/1406.5774

[78] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013. [Online]. Available: http://arxiv.org/abs/1312.6229

[79] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: a review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[80] J. s. Yu, J. Chen, Z. Q. Xiang, and Y. X. Zou, "A hybrid convolutional neural networks with extreme learning machine for wce image classification," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2015, pp. 1822–1827.

[81] M. D. McDonnell, M. D. Tissera, A. van Schaik, and J. Tapson, "Fast, simple and accurate handwritten digit classification using extreme learning machines with shaped input-weights," *CoRR*, vol. abs/1412.8307, 2014. [Online]. Available: http://arxiv.org/abs/1412.8307

[82] S. Pang and X. Yang, "Deep convolutional extreme learning machine and its application in handwritten digit classification," *Intell. Neuroscience*, vol. 2016, pp. 6–, Aug. 2016. [Online]. Available: https://doi.org/10.1155/2016/3049632

[83] W. Zhu, J. Miao, and L. Qing, "Constrained extreme learning machine: A novel highly discriminative random feedforward neural network," in *2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 800–807.

[84] J. Tapson, P. de Chazal, and A. van Schaik, "Explicit computation of input weights in extreme learning machines," *CoRR*, vol. abs/1406.2889, 2014. [Online]. Available: http://arxiv.org/abs/1406.2889

[85] M. D. McDonnell and T. Vladusich, "Enhanced image classification with a fast-learning shallow convolutional neural network," *CoRR*, vol. abs/1503.04596, 2015. [Online]. Available: http://arxiv.org/abs/1503.04596

[86] F. Gürpinar, H. Kaya, H. Dibeklioglu, and A. A. Salah, "Kernel elm and cnn based facial age estimation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2016, pp. 785–791.

[87] Y. Zeng, X. Xu, Y. Fang, and K. Zhao, *Traffic Sign Recognition Using Deep Convolutional Networks and Extreme Learning Machine*. Cham: Springer International Publishing, 2015, pp. 272–280. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23989-7_28

[88] J. Kim, J. Kim, G.-J. Jang, and M. Lee, "Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection," *Neural Networks*, vol. 87, pp. 109 – 121, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608016301885

[89] L. Zhang and D. Zhang, "SVM and ELM: who wins? object recognition with deep convolutional features from imagenet," *CoRR*, vol. abs/1506.02509, 2015. [Online]. Available: http://arxiv.org/abs/1506.02509

[90] L.-l. Cao, W.-b. Huang, and F.-c. Sun, *A Deep and Stable Extreme Learning Approach for Classification and Regression*. Cham: Springer International Publishing, 2015, pp. 141–150. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-14063-6_13

[91] B. Ribeiro and N. Lopes, *Extreme Learning Classifier with Deep Concepts*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 182–189. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41822-8_23

[92] F. Mamalet and C. Garcia, "Simplifying convnets for fast learning," in *Artificial Neural Networks and Machine Learning – ICANN 2012*, A. E. P. Villa, W. Duch, P. Érdi, F. Masulli, and G. Palm, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 58–65.

[93] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[94] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sep. 2015, pp. 1–6.

[95] R. N. Tak, D. M. Agrawal, and H. A. Patil, "Novel phase encoded mel filterbank energies for environmental sound classification," in *Pattern Recognition and Machine Intelligence*, B. U. Shankar, K. Ghosh, D. P. Mandal, S. S. Ray, D. Zhang, and S. K. Pal, Eds. Cham: Springer International Publishing, 2017, pp. 317–325.

[96] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," *CoRR*, vol. abs/1711.10282, 2017. [Online]. Available: http://arxiv.org/abs/1711.10282

[97] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 171–175.

[98] Z. Zhang, S. Xu, S. Cao, and S. Zhang, "Deep convolutional neural network with mixup for environmental sound classification," in *Pattern Recognition and Computer Vision*, J.-H. Lai, C.-L. Liu, X. Chen, J. Zhou, T. Tan, N. Zheng, and H. Zha, Eds. Cham: Springer International Publishing, 2018, pp. 356–367.

[99] Y. Su, K. Zhang, J. Wang, and K. Madani, "Environment sound classification using a two-stream cnn based on decision-level fusion," *Sensors*, vol. 19, no. 7, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1733

[100] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, "Learning attentive representations for environmental sound classification," *IEEE Access*, vol. 7, pp. 130 327–130 339, 2019.

[101] X. Li, V. Chebiyyam, and K. Kirchhoff, "Multi-stream network with temporal attention for environmental sound classification," *CoRR*, vol. abs/1901.08608, 2019. [Online]. Available: http://arxiv.org/abs/1901.08608

[102] J. Sharma, O.-C. Granmo, M. Goodwin, and J. T. Fidje, "Deep convolutional neural networks for fire detection in images," in *Engineering Applications of Neural Networks*. Cham: Springer International Publishing, 2017, pp. 183–193.

[103] J. Sharma, O.-C. Granmo, and M. Goodwin, "Deep cnn-elm hybrid models for fire detection in images," in *Artificial Neural Networks and Machine Learning – ICANN 2018*. Cham: Springer International Publishing, 2018, pp. 245–259.

[104] R. Narayanan, V. M. Lekshmy, S. Rao, and K. Sasidhar, "A novel approach to urban flood monitoring using computer vision," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, July 2014, pp. 1–7.

[105] T. Perol, M. Gharbi, and M. Denolle, "Convolutional neural network for earthquake detection and location," *Science Advances*, vol. 4, no. 2, 2018. [Online]. Available: https://advances.sciencemag.org/content/4/2/e1700578

[106] S. Srivastava, S. Bhugra, B. Lall, and S. Chaudhury, "Drought stress classification using 3d plant models," *CoRR*, vol. abs/1709.09496, 2017. [Online]. Available: http://arxiv.org/abs/1709.09496

[107] A. U. Waldeland, J. H. Reksten, and A.-B. Salberg, "Avalanche detection in sar images using deep learning," *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2386–2389, 2018.

[108] L. Lopez-Fuentes, J. van de Weijer, M. G. Hidalgo, H. Skinnemoen, and A. D. Bagdanov, "Review on computer vision techniques in emergency situation," *CoRR*, vol. abs/1708.07455, 2017. [Online]. Available: http://arxiv.org/abs/1708.07455

[109] J. D. Fair, W. F. Bailey, R. A. Felty, A. E. Gifford, B. Shultes, and L. H. Volles, "Quantitation by portable gas chromatography: mass spectrometry of vocs associated with vapor intrusion," *International journal of analytical chemistry*, vol. 2010, 2010.

[110] I. Mocanu and A. M. Florea, "A model for activity recognition and emergency detection in smart environments," in *The First International Conference on Ambient Computing, Applications, Services and Technologies*, 2011, pp. 23–29.

[111] A. Mao, X. Ma, Y. He, and J. Luo, "Highly portable, sensor-based system for human fall monitoring," *Sensors*, vol. 17, no. 9, p. 2096, 2017.

[112] Y. Liu, E. Racah, Prabhat, J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. F. Wehner, and W. D. Collins, "Application of deep convolutional neural networks for detecting extreme weather in climate datasets," *CoRR*, vol. abs/1605.01156, 2016. [Online]. Available: http://arxiv.org/abs/1605.01156

[113] M. Nadjafi, M. A. Farsi, and H. Jabbari, "Reliability analysis of multi-state emergency detection system using simulation approach based on fuzzy

failure rate," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 3, pp. 532–541, Sep 2017. [Online]. Available: https://doi.org/10.1007/s13198-016-0563-7

[114] M. Dener, Y. Ozkok, and C. Bostancioglu, "Fire detection systems in wireless sensor networks," *Procedia - Social and Behavioral Sciences*, vol. 195, pp. 1846 – 1850, 2015, world Conference on Technology, Innovation and Entrepreneurship. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042815038872

[115] J. Zhang, W. Li, Z. Yin, S. Liu, and X. Guo, "Forest fire detection system based on wireless sensor network," in *2009 4th IEEE Conference on Industrial Electronics and Applications*, May 2009, pp. 520–523.

[116] A. Khadivi and M. Hasler, "Fire detection and localization using wireless sensor networks," in *Sensor Applications, Experimentation, and Logistics*, N. Komninos, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 16–26.

[117] Q. Nguyen, S. Yun, and J. Choi, "Detection of audio-based emergency situations using perception sensor network," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Aug 2016, pp. 763–766.

[118] D. K. Fragoulis and J. Avaritsiotis, "A siren detection system based on mechanical resonant filters," *Sensors*, vol. 1, 09 2001.

[119] T. Miyazaki, Y. Kitazono, and M. Shimakawa, "Ambulance siren detector using fft on dspic," in *ICIS 2013*, 2013.

[120] L. Marchegiani and P. Newman, "Listening for sirens: Locating and classifying acoustic alarms in city scenes," *CoRR*, vol. abs/1810.04989, 2018. [Online]. Available: http://arxiv.org/abs/1810.04989

[121] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[122] M. S. Ebrahimi and H. K. Abadi, "Study of residual networks for image recognition," *CoRR*, vol. abs/1805.00325, 2018. [Online]. Available: http://arxiv.org/abs/1805.00325

[123] A. E. Orhan, "Skip connections as effective symmetry-breaking," *CoRR*, vol. abs/1701.09175, 2017. [Online]. Available: http://arxiv.org/abs/1701.09175

[124] T. W. Megan Reeve and B. Altevogt, *Improving Data Collection Capabilities and Information Resources*. National Academies Press (US), April 2015.

[125] M. O. Columb, P. Haji-Michael, and P. Nightingale, "Data collection in the emergency setting," *Emergency Medicine Journal*, vol. 20, no. 5, pp. 459–463, 2003. [Online]. Available: https://emj.bmj.com/content/20/5/459

[126] R. P. Susana Arroyo Barrantes, Martha Rodriguez, Ed., *Information management and communication in emergencies and disasters*. World Health Organization (WHO), 2009.

[127] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017.

[128] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[129] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[130] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[131] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

[132] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[133] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[134] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[135] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *CoRR*, vol. abs/1809.02627, 2018. [Online]. Available: http://arxiv.org/abs/1809.02627

[136] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. P. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, "Starcraft II: A new challenge for reinforcement learning," *CoRR*, vol. abs/1708.04782, 2017. [Online]. Available: http://arxiv.org/abs/1708.04782

[137] P. Shinners, "Pygame," http://pygame.org/, 2011.

[138] Ł. Kidziński, S. P. Mohanty, C. Ong, J. Hicks, S. Francis, S. Levine, M. Salathé, and S. Delp, "Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning," in *NIPS 2017 Competition Book*, S. Escalera and M. Weimer, Eds. Springer: Springer, 2018.

[139] A. Abdolmaleki, M. Movahedi, S. Salehi, N. Lau, and L. P. Reis, "A reinforcement learning based method for optimizing the process of decision making in fire brigade agents," in *Progress in Artificial Intelligence*, L. Antunes and H. S. Pinto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 340–351.

[140] F. Pahlevan Aghababa, M. Shimizu, F. Amigoni, A. Kabiri, and A. Visser, "Robocup 2018 robocup rescue simulation league virtual robot competition rules document," May 2018.

[141] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge University Engineering Department, Cambridge, England, Tech. Rep. TR 166, 1994.

[142] D. Moura and E. Oliveira, “Fighting fire with agents: An agent coordination model for simulated firefighting,” in *Proceedings of the 2007 Spring Simulation Multiconference - Volume 2*, ser. SpringSim ’07. San Diego, CA, USA: Society for Computer Simulation International, 2007, pp. 71–78. [Online]. Available: http://dl.acm.org/citation.cfm?id=1404680.1404691

[143] H. Zhao and S. Winter, “A time-aware routing map for indoor evacuation,” *Sensors*, vol. 16, no. 1, 2016.

[144] A. Wharton, “Simulation and investigation of multi-agent reinforcement learning for building evacuation scenarios *,” 2009. [Online]. Available: https://pdfs.semanticscholar.org/08b9/4add93ca0ef8f6dd40ad78a6a988c514a4d8.pdf

[145] M. L. Chu, P. Parigi, K. Law, and J.-C. Latombe, “Modeling social behaviors in an evacuation simulator,” *Computer Animation and Virtual Worlds*, vol. 25, no. 3-4, pp. 373–382. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1595

[146] V. J. Cassol, E. S. Testa, C. R. Jung, M. Usman, P. Faloutsos, G. Berseth, M. Kapadia, N. I. Badler, and S. R. Musse, “Evaluating and optimizing evacuation plans for crowd egress,” *IEEE Computer Graphics and Applications*, vol. 37, no. 4, pp. 60–71, 2017.

[147] L. Lee and Y.-J. Son, “Dynamic learning in human decision behavior for evacuation scenarios under bdi framework,” in *Proceedings of the 2009 INFORMS Simulation Society Research Workshop. INFORMS Simulation Society: Catonsville, MD*, 2009, pp. 96–100.

[148] S. Lee, Y.-J. Son, and J. Jin, “An integrated human decision making model for evacuation scenarios under a bdi framework,” *ACM Trans. Model. Comput. Simul.*, vol. 20, no. 4, pp. 23:1–23:24, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1842722.1842728

[149] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML ’09. New York, NY, USA: ACM, 2009, pp. 41–48. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553380

[150] Y. Wang, Q. Meng, W. Chen, Y. Liu, Z. Ma, and T. Liu, “Target transfer q-learning and its convergence analysis,” *CoRR*, vol. abs/1809.08923, 2018. [Online]. Available: http://arxiv.org/abs/1809.08923

[151] C. Szepesvári, "The asymptotic convergence-rate of q-learning," in *Advances in Neural Information Processing Systems*, 1998, pp. 1064–1070.

[152] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[153] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015. [Online]. Available: http://arxiv.org/abs/1502.03044

[154] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. [Online]. Available: http://arxiv.org/abs/1409.0473

[155] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[156] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[157] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 1041–1044. [Online]. Available: http://doi.acm.org/10.1145/2647868.2655045

[158] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," 2015.

[159] *MATLAB Signal Processing Toolbox 2019*. Natick, Massachusetts, United States: The MathWorks Inc., 2019.

[160] M. Thoma, "A survey of semantic segmentation," *CoRR*, vol. abs/1602.06541, 2016. [Online]. Available: http://arxiv.org/abs/1602.06541

[161] M. Abadi, A. Agarwal, P. Barham, and Others, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[162] P. Yakubovskiy, “Segmentation models,” https://github.com/qubvel/segmentation_models, 2019.

[163] A. B. Jung, “imgaug,” https://github.com/aleju/imgaug, 2018, [Online; accessed 30-Oct-2018].

[164] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[165] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[166] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results,” http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[167] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudik, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[168] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[169] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[170] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063. [Online]. Available: http://dl.acm.org/citation.cfm?id=3009657.3009806

[171] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 1928–1937. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045594

[172] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[173] Y. Wu, E. Mansimov, S. Liao, A. Radford, and J. Schulman, "Openai baselines: Acktr and a2c," https://openai.com/blog/baselines-acktr-a2c/, 2017.

[174] Y. Wu, E. Mansimov, S. Liao, R. B. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *CoRR*, vol. abs/1708.05144, 2017. [Online]. Available: http://arxiv.org/abs/1708.05144

[175] J. Martens and R. B. Grosse, "Optimizing neural networks with kronecker-factored approximate curvature," *CoRR*, vol. abs/1503.05671, 2015. [Online]. Available: http://arxiv.org/abs/1503.05671

[176] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, p. 14, 2013.

PART II

Paper A

Title:	Deep Convolutional Neural Networks for Fire Detection in Images
Authors:	Jivitesh Sharma, Ole-Christoffer Granmo, Morten Goodwin and Jahn Thomas Fidje
Affiliation:	Dept. of Information and Communication Technology, University of Agder (UiA), Grimstad, Norway
Conference:	18th *International Conference on Engineering Applications of Neural Networks (EANN)*, Aug. 2017, Athens, Greece.
Copyright ©:	Springer

Deep Convolutional Neural Networks for Fire Detection in Images

Jivitesh Sharma, Ole-Christoffer Granmo, Morten Goodwin and Jahn Thomas Fidje

University of Agder (UiA), Norway

Abstract. Detecting fire in images using image processing and computer vision techniques has gained a lot of attention from researchers during the past few years. Indeed, with sufficient accuracy, such systems may outperform traditional fire detection equipment. One of the most promising techniques used in this area is Convolutional Neural Networks (CNNs). However, the previous research on fire detection with CNNs has only been evaluated on balanced datasets, which may give misleading information on real-world performance, where fire is a rare event. Actually, as demonstrated in this paper, it turns out that a traditional CNN performs relatively poorly when evaluated on the more realistically balanced benchmark dataset provided in this paper. We therefore propose to use even deeper Convolutional Neural Networks for fire detection in images, and enhancing these with fine tuning based on a fully connected layer. We use two pretrained state-of-the-art Deep CNNs, VGG16 and Resnet50, to develop our fire detection system. The Deep CNNs are tested on our imbalanced dataset, which we have assembled to replicate real world scenarios. It includes images that are particularly difficult to classify and that are deliberately unbalanced by including significantly more non-fire images than fire images. The dataset has been made available online. Our results show that adding fully connected layers for fine tuning indeed does increase accuracy, however, this also increases training time. Overall, we found that our deeper CNNs give good performance on a more challenging dataset, with Resnet50 slightly outperforming VGG16. These results may thus lead to more successful fire detection systems in practice.

Keywords: Fire Detection, Deep Convolutional Neural Networks, VGG16, Resnet50.

1 Introduction

Emergency situations like floods, earthquakes and fires pose a big threat to public health and safety, property and environment. Fire related disasters are the most common type of Emergency situation which requires thorough analysis of the situation required for a quick and precise response. The first step involved in this process is to detect fire in the environment as quickly and accurately as

possible.
Fire Detection in most places employs equipment like temperature detectors, smoke detectors, thermal cameras etc. which is expensive and not available to all [14]. But, after the advent of advanced image processing and computer vision techniques, detection of fire may not require any equipment other than cameras. Due to this expeditious development in vision-based fire detection models, there is a particular inclination towards replacing the traditional fire detection tools with vision-based models. These models have many advantages over their hardware based counterparts like accuracy, more detailed view of the situation, less prone to errors, robustness towards the environment, considerably lower cost and the ability to work on existing camera surveillance systems.
There have been many innovative techniques proposed in the past to build an accurate fire detection system which are broadly based on image processing and computer vision techniques. The state-of-the-art vision-based techniques for fire and smoke detection have been comprehensively evaluated and compared in [21]. The colour analysis technique has been widely used in the literature to detect and analyse fire in images and videos [2, 13, 16, 20]. On top of colour analysis, many novel methods have been used to extract high level features from fire images like texture analysis [2], dynamic temporal analysis with pixel-level filtering and spatial analysis with envelope decomposition and object labelling [22], fire flicker and irregular fire shape detection with wavelet transform [20], etc. These techniques give adequate performance but are outperformed by Machine Learning techniques. A comparative analysis between colour-based models for extraction of rules and a Machine Learning algorithm is done for the fire detection problem in [19]. The machine learning technique used in [19] is Logistic Regression which is one of the simplest techniques in Machine Learning and still outperforms the colour-based algorithms in almost all scenarios. These scenarios consist of images containing different fire pixel colours of different intensities, with and without smoke.
Instead of using many different algorithms on top of each other to extract relevant features, we can use a network that learns relevant features on its own. Neural networks have been successfully used in many different areas such as Natural Language Processing, Speech Recognition, Text Analysis and especially Image Classification. Extracting relevant features from images is the key to accurate classification and analysis which is why the problem of fire detection is ideally suited for Deep Learning. Deep Neural Networks are used to automatically 'learn' hierarchy of pertinent features from data without human intervention and the type of neural network ideally suited for image classification is the Convolutional Neural Networks (CNN).
Therefore, our approach is to employ state-of-the-art CNNs to distinguish between images that containing fire and images that do not and build an accurate fire detection system. To make these models more robust, we use a custom-made image dataset containing images with numerous scenarios.
The rest of paper is organised in the following manner: Section 2 briefly describes the previous research that uses CNNs for detecting fire. In Section 3 give a de-

scription of our proposed work. Section 4 gives the experimental results along with an illustration of our dataset, which is available online for the research community. Finally, Section 5 concludes our paper.

2 Related Work

There have been many significant contributions from various researchers in developing a system that can accurately detect fire in the surrounding environment. But, the most notable research in this field involves Deep Convolutional Neural Networks (DCNN). DCNN models are currently among the most successful image classification models which makes them ideal for a task such as Fire detection in images. This has been demonstrated by previous research published in this area.
In [5], the authors use CNN for detection of fire and smoke in videos. A simple sequential CNN architecture, similar to LeNet-5 [11], is used for classification. The authors quote a testing accuracy of 97.9% with a satisfactory false positive rate.
Whereas in [23], a very innovative cascaded CNN technique is used to detect fire in an image, followed by fine-grained localisation of patches in the image that contain the fire pixels. The cascaded CNN consists of AlexNet CNN architecture [10] with pre-trained ImageNet weights [15] and another small network after the final pooling layer which extracts patch features and labels the patches which contain fire. Different patch classifiers are compared.
The AlexNet architecture is also used in [18] which is used to detect smoke in images. It is trained on a fairly large dataset containing smoke and non-smoke images for a considerably long time. The quoted accuracies for large and small datasets are 96.88% and 99.4% respectively with relatively low false positive rates.
Another paper that uses the AlexNet architecture is [12]. This paper builds its own fire image and video dataset by simulating fire in images and videos using Blender. It adds fire to frames by adding fire properties like shadow, fore-ground fire, mask etc. separately. The animated fire and video frames are composited using OpenCV [1]. The model is tested on real world images. The results show reasonable accuracy with high false positive rate.
As opposed to CNNs which extract features directly from raw images, in some methods image/video features are extracted using image processing techniques and then given as input to a neural network. Such an approach has been used in [4]. The fire regions from video frames are obtained by threshold values in the HSV colour space. The general characteristics of fire are computed using these values from five continuous frames and their mean and standard deviation is given as input to a neural network which is trained using back propagation to identify forest fire regions. This method performs segmentation of images very accurately and the results show high accuracy and low false positive rates.
In [8], a neural network is used to extract fire features based on the HSI colour model which gives the fire area in the image as output. The next step is fire area

segmentation where the fire areas are roughly segmented and spurious fire areas like fire shadows and fire-like objects are removed by image difference. After this the change in shape of fire is estimated by taking contour image difference and white pixel ratio to estimate the burning degree of fire, i.e. no-fire, small, medium and large. The experimental results show that the method is able to detect different fire scenarios with relatively good accuracy.
All the research work done in this area has been exemplary. But, there are some issues associated with each of them that we try to alleviate in this paper. We use a dataset that consists of images that we have handpicked from the internet. The dataset contains images that are extremely hard to classify which results in poor generalization. The dataset also contains many different scenarios and is highly unbalanced to replicate real world behaviour. In this paper, we propose to use state-of-the-art pre-trained DCNN models. The reason behind using such complex models is explained in the next section. We also modify these models to improve accuracy at the cost of training time.

3 The Fire Detector

In this paper, we propose to employ Deep Convolutional Neural Networks instead of simple and shallow CNN models. The AlexNet has been used by researchers in the past for fire detection which has produced satisfactory results. We propose to use two Deep CNN architectures that have outperformed the AlexNet on the ImageNet dataset, namely VGG16 [17] and Resnet50 [7]. We use these models with pre-trained ImageNet weights. This helps greatly when there is lack of training data. So, we just have to fine-tune the fully-connected layers on our dataset.

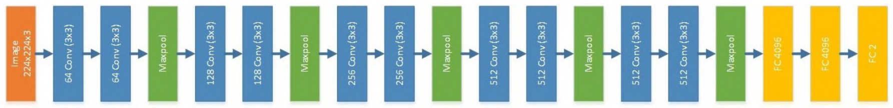
3.1 Deep ConvNet Models

The Convolutional Neural Network was first introduced in 1980 by Kunihiko Fukushima [6]. The CNN is designed to take advantage of two dimensional structures like 2D Images and capture local spatial patterns. This is achieved with local connections and tied weights. It consists of one or more convolution layers with pooling layers between them, followed by one or more fully connected layers, as in a standard multilayer perceptron. CNNs are easier to train compared to Deep Neural Networks because they have fewer parameters and local receptive fields.
In CNNs, kernels/filters are used to see where particular features are present in an image by convolution with the image. The size of the filters gives rise to locally connected structure which are each convolved with the image to produce feature maps. The feature maps are usually subsampled using mean or max pooling. The reduction is parameters is due to the fact that convolution layers share weights. The reason behind parameter sharing is that we make an assumption, that the statistics of a patch of a natural image are the same as any other patch of the image, which suggests that features learned at a location can also be learned for

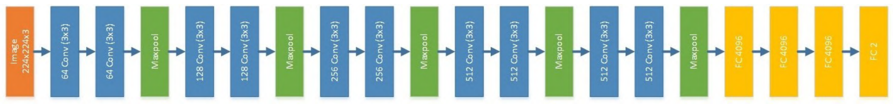
other locations. So, we can apply this learned feature detector anywhere in the image. This makes CNNs ideal feature extractors for images.
The CNNs with many layers have been used for various applications especially image classification. In this paper, we use two state-of-the-art Deep CNNs that have achieved one of the lowest errors in image classification tasks.

VGG16: The VGG16 architecture was proposed by the Visual Geometry Group at the University of Oxford [17]. The main purpose of the paper was to investigate the effect of depth in CNN models. They developed a number of models with different depths ranging from 11 layers to 19 layers and tested them on different tasks. The results on these tasks show that increasing depth also increases performance and accuracy. The 19 layer architecture, VGG19 won the ImageNet challenge in 2014, but the 16 layer architecture, VGG16 achieved an accuracy which was very close to VGG19. Both the models are simple and sequential. The 3x3 convolution filters are used in the VGG models which is the smallest size and thus captures local features. The 1x1 convolutions can be viewed as linear transformations and can also be used for dimensionality reduction. We choose the VGG16 over the VGG19 because it takes less time to train and the classification task in hand is not as complex as ImageNet challenge. Both the models have the same number of fully connected layers, i.e. 3, but differ in the number of 3x3 filters.

VGG16 (modified): In this work, we also test a modified version of VGG16 which consists of 4 fully connected layers, fine-tuned on the training data, which was able to increase the accuracy of classification. We also tested with more fully connected layers but the increase in accuracy was overshadowed by the increase in training time. The figures 1(a) and 1(b) show the original and modified VGG16 architectures respectively.



(a) VGG16 Architecture



(b) Modified VGG16 Architecture

Fig. 1

Resnet50: After the success of the VGG architectures, it was established that deeper models outperform shallower networks. But, the problem with making models deeper was the difficulty in training them because model complexity increases as the number of layers increase. This issue was addressed by Microsoft Research, who proposed extremely deep architectures but with lower complexity [7]. They introduced a new framework of learning to ease training of such deep networks. This is called Residual learning and hence the models that employed this framework are called Residual Networks. Residual Learning involves learning residual functions. If a few stacked layers can approximate a complex function, $F(x)$ where, x is the input to the first layer, then they can also approximate the residual function $F(x) - x$. So, instead the stacked layers approximate the residual function$G(x) = F(x) - x$, where the original function becomes $G(x) + x$. Even though both can capable of approximating the desired function, the ease of training with residual functions is better. These residual functions are forwarded across layers in the network using identity mapping shortcut connections. The ImageNet 2015 results show that Resnet achieves the lowest error rates in image classification. The Resnet architectures consist of networks of various depths: 18-layers, 34-layers, 50-layers, 101-layers and 152-layers. We choose the architecture with intermediate depth, i.e. 50 layers. The Resnet consists of 3x3 and 1x1 filters, pooling layers and residual connections and a single softmax layer at the end.

Resnet50 (modified): We also test a modified Resnet model by adding a fully connected layer fine-tuned on the training data, which increase accuracy further. We did not add any more fully connected layers since the model is already quite deep and takes a long time to train. The figures 2(a) and 2(b) show the original and modified Resnet50 architectures respectively.

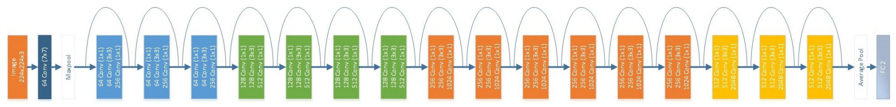


Image 224x224x3
64 Conv (7x7)
Maxpool
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
Average Pool
FC 2

(a) Resnet50 Architecture

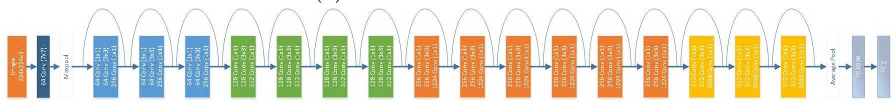


Image 224x224x3
64 Conv (7x7)
Maxpool
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
64 Conv (1x1) 64 Conv (3x3) 256 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
128 Conv (1x1) 128 Conv (3x3) 512 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
256 Conv (1x1) 256 Conv (3x3) 1024 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
512 Conv (1x1) 512 Conv (3x3) 2048 Conv (1x1)
Average Pool
FC 4096
FC 2

(b) Modified Resnet50 Architecture

Fig. 2

4 Experiments

We conducted our experiments to compare training and testing accuracies and execution times of the VGG16 and Resnet50 models including modifications.

We also trained a simple CNN which is used in [5] and compare with much deeper models to show why deeper and more complex models are necessary for fire detection on our dataset. We also train the modified VGG16 and Resnet50 models and compare the performance. We used pre-trained Keras [3] models and fine-tuned the fully-connected layers on our dataset. The training of the models was done on the following hardware specifications: Intel i5 2.5GHz, 8GB RAM and Nvidia Geforce GTX 820 2GB GPU. Each model was trained on the dataset for 10 training epochs with the ADAM optimizer [9] with default parameters $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The details of the dataset are given in the next subsection.

4.1 The Dataset

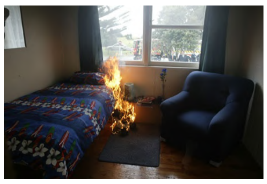
Since there is no benchmark dataset for fire detection in images, we created our own dataset by handpicking images from the internet. [1]This dataset consists of 651 images which is quite small in size but it enables us to test the generalization capabilities and the effectiveness and efficiency of models to extract relevant features from images when training data is scarce. The dataset is divided into training and testing sets. The training set consists of 549 images: 59 fire images and 490 non-fire images. The imbalance is delibrate to replicate real world situations, as the probability of occurrence of fire hazards is quite small. The datasets used in previous papers have been balanced which does not imitate the real world environment. The testing set contains 102 images: 51 images each of fire and non-fire classes. As the training set is highly unbalanced and the testing set is exactly balanced, it makes a good test to see whether the models are able to generalize well or not. For a model with good accuracy, it must be able to extract the distinguishing features from the small amount of fire images. To extract such features from small amount of data the model must be deep enough. A poor model would just label all images as non-fire, which is the case shown in the results.
Apart from being unbalanced, there are a few images that are very hard to classify. The dataset contains images from all scenarios like fire in a house, room, office, forest fire, with different illumination intensity and different shades of red, yellow and orange, small and big fires, fire at night, fire in the morning; non-fire images contain a few images that are hard to distinguish from fire images like a bright red room with high illumination, sunset, red coloured houses and vehicles, bright lights with different shades of yellow and red etc.
The figures 3(a) to 3(f) show the fire images with different environments: indoor, outdoor, daytime, nighttime, forest fire, big and small fire. And the figures 4(a) to 4(f) show the non-fire images that are difficult to classify. Considering these characteristics of our dataset, detecting fire can be a difficult task. We have made the dataset available online so that it can be used for future research in this area.

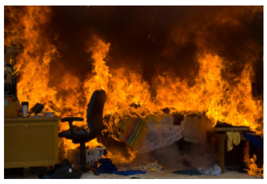
[1] The dataset is available here: `https://github.com/UIA-CAIR/Fire-Detection-Image-Dataset`

(a)

(b)

(c)


(d)


(e)

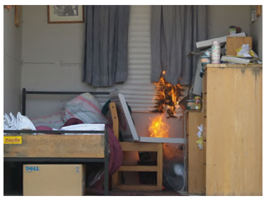
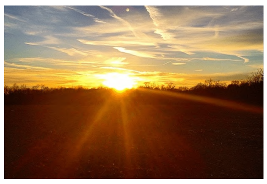
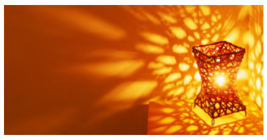
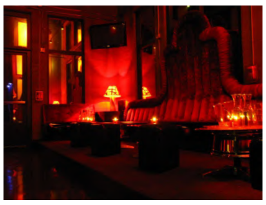
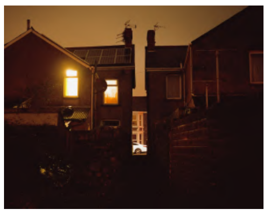

(f)

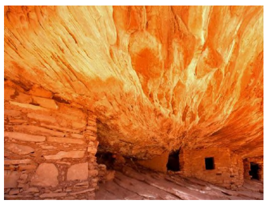
Fig. 3: Examples of Fire Images


(a)


(b)


(c)


(d)


(e)

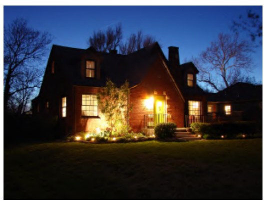

(f)

Fig. 4: Examples of Non-Fire Images that are difficult to classify

4.2 Results

Table 1. shows the results of our experiments. The simple CNN model labels all images as non-fire which means that it is unable to extract relevant features from the dataset and cannot handle unbalanced datasets, which we can see from the training accuracy which is exactly equal to the percentage of non-fire images in the training set. So, the simple CNN overfits on the majority class of the unbalanced training dataset. Since, the training and fine-tuning methods for all models used here are the same, at the end it comes down to the architecture of the model. This justifies the use of deeper models like VGG16 and Resnet50. The simple CNN tested on our dataset is similar to the one used in [5]. The deep

Table 1: Comparison between CNN models

Model	Training accuracy	Training time (in sec)	Testing accuracy	Testing time (in sec)
VGG16	100	7149	90.19	121
VGG16 (modified)	100	7320	91.18	122
Resnet50	100	15995	91.18	105
Resnet50 (modified)	100	16098	92.15	107
Simple CNN [5]	89.25	112	50.00	2

models achieve testing accuracy greater than 90%. And, the modified VGG16 and Resnet50 models outperform the base models by a small margin with slightly higher training time. It seems obvious that adding fully-connected layers to a network would increase accuracy. But on such a small dataset, the trade-off between accuracy and training time is quite poor, so we stop after adding just one fully connected layer. We also tested for more fully-connected layers(which is feasible since the model is pre-trained) but the increase in accuracy compared to increase in training time was too small.
Overall, the deep models perform well on this dataset. This shows that these models generalize well even when there is lack of training data. This means that if we want to slightly alter what the model does, we do not require large amount of data for retraining.

5 Conclusion

In this paper, we have proposed to use two state-of-the-art Deep Convolutional Neural Networks for fire detection in images, VGG16 and Resnet50. We test these models on our dataset which is made specifically to replicate real world environment. The dataset includes images that are difficult to classify and is highly unbalanced by including less fire images and more non-fire images since fire is a rare occurrence in the real world. We rationalize the use of such deep and complex models by showing that a simple CNN performs poorly on our dataset.

To further increase accuracy, we added fully connected layers to both VGG16 and Resnet50. Results show that adding fully connected layers does improve the accuracy of the detector but also increases its training time. In practice, increasing the number of fully connected layers by more than one results in minute increase in accuracy compared to the large increase in training time, even if the models are pre-trained. To conclude, we found that deep CNNs provide good performance on a diverse and highly imbalanced dataset of small size, with Resnet50 slightly outperforming VGG16 and adding fully connected layers slightly improves accuracy but takes longer to train.

References

1. G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.
2. Daniel Yoshinobu Takada Chino, Letricia P. S. Avalhais, José Fernando Rodrigues Jr., and Agma J. M. Traina. Bowfire: Detection of fire in still images by integrating pixel color and texture analysis. *CoRR*, abs/1506.03495, 2015.
3. Francois Chollet. Keras, 2015.
4. J. Zhao Z. Zhang C. Qu Y. Ke D. Zhang, S. Han and X. Chen. Image based forest fire detection using dynamic characteristics with artificial neural networks. In *2009 International Joint Conference on Artificial Intelligence*, pages 290–293, April 2009.
5. S. Frizzi, R. Kaabi, M. Bouchouicha, J. M. Ginoux, E. Moreau, and F. Fnaiech. Convolutional neural network for video fire and smoke detection. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 877–882, Oct 2016.
6. Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
7. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
8. Wen-Bing Horng and Jian-Wen Peng. Image-based fire detection using neural networks. In *JCIS*, 2006.
9. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
10. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
11. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
12. Bc. Tomas Polednik. Detection of fire in images and video using cnn. *Excel@FIT*, 2015.
13. K. Poobalan and S.C. Liew. Fire detection algorithm using image processing techniques. In *3rd International Conference on Artificial Intelligence and Computer Science (AICS2015)*, Ocotober 2015.
14. Richard Bright Richard Custer. Fire detection: The state of the art. *NBS Technical Note, US Department of Commerce*, 1974.

15. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
16. Jing Shao, Guanxiang Wang, and Wei Guo. An image-based fire detection method using color analysis. In *2012 International Conference on Computer Science and Information Processing (CSIP)*, pages 1008–1011, Aug 2012.
17. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
18. C.Tao, J.Zhang, and P.Wang. Smoke detection based on deep convolutional neural networks. In *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, pages 150–153, Dec 2016.
19. Tom Toulouse, Lucile Rossi, Turgay Celik, and Moulay Akhloufi. Automatic fire pixel detection using image processing: a comparative analysis of rule-based and machine learning-based methods. *Signal, Image and Video Processing*, 10(4):647–654, 2016.
20. B.Ugur Toreyin, Yigithan Dedeoglu, Ugur Gudukbay, and A.Enis Cetin. Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters*, 27(1):49 – 58, 2006.
21. Steven Verstockt, Peter Lambert, Rik Van de Walle, Bart Merci, and Bart Sette. State of the art in vision-based fire and smoke dectection. In Heinz Luck and Ingolf Willms, editors, *International Conference on Automatic Fire Detection, 14th, Proceedings*, volume 2, pages 285–292. University of Duisburg-Essen. Department of Communication Systems, 2009.
22. Jerome Vicente and Philippe Guillemant. An image processing technique for automatically detecting forest fire. *International Journal of Thermal Sciences*, 41(12):1113 – 1120, 2002.
23. Qingjie Zhang, Jiaolong Xu, Liang Xu, and Haifeng Guo. Deep convolutional neural networks for forest fire detection. February 2016.

Paper B

Title:	Deep CNN-ELM Hybrid Models for Fire Detection in Images
Authors:	Jivitesh Sharma, Ole-Christoffer Granmo and Morten Goodwin
Affiliation:	Dept. of Information and Communication Technology, University of Agder (UiA), Grimstad, Norway
Conference:	27th *International Conference on Artificial Neural Networks*, Oct. 2018, Rhodes, Greece.
Copyright ©:	Springer

Deep CNN-ELM Hybrid models for Fire Detection in Images

Jivitesh Sharma, Ole-Christopher Granmo, and Morten Goodwin

Center for Artificial Intelligence Research, University of Agder,
Jon Lilletuns vei 9, 4879 Grimstad, Norway
`jivitesh.sharma@uia.no`
`ole.granmo@uia.no`
`morten.goodwin@uia.no`

Abstract. In this paper, we propose a hybrid model consisting of a Deep Convolutional feature extractor followed by a fast and accurate classifier, the Extreme Learning Machine, for the purpose of fire detection in images. The reason behind using such a model is that Deep CNNs used for image classification take a very long time to train. Even with pre-trained models, the fully connected layers need to be trained with backpropagation, which can be very slow. In contrast, we propose to employ the Extreme Learning Machine (ELM) as the final classifier trained on pre-trained Deep CNN feature extractor. We apply this hybrid model on the problem of fire detection in images. We use state of the art Deep CNNs: VGG16 and Resnet50 and replace the softmax classifier with the ELM classifier. For both the VGG16 and Resnet50, the number of fully connected layers is also reduced. Especially in VGG16, which has 3 fully connected layers of 4096 neurons each followed by a softmax classifier, we replace two of these with an ELM classifier. The difference in convergence rate between fine-tuning the fully connected layers of pre-trained models and training an ELM classifier are enormous, around 20x to 51x speed-up. Also, we show that using an ELM classifier increases the accuracy of the system by 2.8% to 7.1% depending on the CNN feature extractor. We also compare our hybrid architecture with another hybrid architecture, i.e. the CNN-SVM model. Using SVM as the classifier does improve accuracy compared to state-of-the-art deep CNNs. But our Deep CNN-ELM model is able to outperform the Deep CNN-SVM models. [1]

Keywords: Deep Convolutional Neural Networks, Extreme Learning Machine, Image Classification, Fire Detection

1 Introduction

The problem of fire detection in images has received a lot of attention in the past by researchers from computer vision, image processing and deep learning.

[1] Preliminary version of some of the results of this paper appear in Deep Convolutional Neural Networks for Fire Detection in Images, Springer Proceedings Engineering Applications of Neural Networks 2017 (EANN'17), Athens, Greece, 25-27 August

This is a problem that needs to be solved without any compromise. Fire can cause massive and irrevocable damage to health, life and property. It has led to over a 1000 deaths a year in the US alone, with property damage in access of one billion dollars. Besides, the fire detectors currently in use require different kinds of expensive hardware equipment for different types of fire [27].
What makes this problem even more interesting is the changing background environment due to varying luminous intensity of the fire, fire of different shades, different sizes etc. Also, the false alarms due to the environment resembling fire pixels, like room with bright red/orange background and bright lights. Furthermore, the probability of occurrence of fire is quite low, so the system must be trained to handle imbalance classification.
Various techniques have been used to classify between images that contain fire and images that do not. The state-of-the-art vision-based techniques for fire and smoke detection have been comprehensively evaluated and compared in [39]. The colour analysis technique has been widely used in the literature to detect and analyse fire in images and videos [4, 24, 31, 37]. On top of colour analysis, many novel methods have been used to extract high level features from fire images like texture analysis [4], dynamic temporal analysis with pixel-level filtering and spatial analysis with envelope decomposition and object labelling [40], fire flicker and irregular fire shape detection with wavelet transform [37], etc.
These techniques give adequate performance but are currently outperformed by Machine Learning techniques. A comparative analysis between colour-based models for extraction of rules and a Machine Learning algorithm is done for the fire detection problem in [36]. The machine learning technique used in [36] is Logistic Regression which is one of the simplest techniques in Machine Learning and still outperforms the colour-based algorithms in almost all scenarios. These scenarios consist of images containing different fire pixel colours of different intensities, with and without smoke.
Instead of explicitly designing features by using image processing techniques, deep neural networks can be used to extract and learn relevant features from images. The Convolutional Neural Networks (CNNs) are the most suitable choice for the task of image processing and classification.
In this paper, we employ state-of-the-art Deep CNNs for fire detection and then propose to use hybrid CNN-ELM and CNN-SVM models to outperform Deep CNNs. Such hybrid models have been used in the past for image classification, but the novelty of our approach lies in using state-of-the-art Deep CNNs like VGG16 and Resnet50 as feature extractors and then remove some/all fully connected layers with an ELM classifier. This models outperform Deep CNNs in terms of accuracy, training time and size of the network. We also compare the CNN-ELM model with another hybrid model, CNN-SVM and show that the CNN-ELM model gives the best performance.
The rest of the paper is organized in the following manner: Section 2 briefly describes the related work with CNNs for fire detection and Hybrid models for image classification. Section 3 explains our work in detail and section 4 gives

details of our experiments and presents the results. Section 5 summarizes and concludes our work.

2 Related Work

In this paper, we integrate state-of-the-art CNN hybrid models and apply it to the problem of fire detection in images. To the best of our knowledge, hybrid models have never been applied to fire detection. So, we present a brief overview of previous research done in CNNs used for fire detection and hybrid models separately in the next two sub-sections.

2.1 CNNs for Fire detection

There have been many significant contributions from various researchers in developing a system that can accurately detect fire in the surrounding environment. But, the most notable research in this field involves Deep Convolutional Neural Networks (Deep CNN). Deep CNN models are currently among the most successful image classification models which makes them ideal for a task such as Fire detection in images. This has been demonstrated by previous research published in this area.

In [7], the authors use CNN for detection of fire and smoke in videos. A simple sequential CNN architecture, similar to LeNet-5 [18], is used for classification. The authors quote a testing accuracy of 97.9% with a satisfactory false positive rate.

Whereas in [43], a very innovative cascaded CNN technique is used to detect fire in an image, followed by fine-grained localisation of patches in the image that contain the fire pixels. The cascaded CNN consists of AlexNet CNN architecture [17] with pre-trained ImageNet weights [28] and another small network after the final pooling layer which extracts patch features and labels the patches which contain fire. Different patch classifiers are compared.

The AlexNet architecture is also used in [34] which is used to detect smoke in images. It is trained on a fairly large dataset containing smoke and non-smoke images for a considerably long time. The quoted accuracies for large and small datasets are 96.88% and 99.4% respectively with relatively low false positive rates.

Another paper that uses the AlexNet architecture is [23]. This paper builds its own fire image and video dataset by simulating fire in images and videos using Blender. It adds fire to frames by adding fire properties like shadow, fore-ground fire, mask etc. separately. The animated fire and video frames are composited using OpenCV [2]. The model is tested on real world images. The results show reasonable accuracy with high false positive rate.

As opposed to CNNs which extract features directly from raw images, in some methods image/video features are extracted using image processing techniques and then given as input to a neural network. Such an approach has been used in [6]. The fire regions from video frames are obtained by threshold values in the

HSV colour space. The general characteristics of fire are computed using these values from five continuous frames and their mean and standard deviation is given as input to a neural network which is trained using back propagation to identify forest fire regions. This method performs segmentation of images very accurately and the results show high accuracy and low false positive rates.
In [11], a neural network is used to extract fire features based on the HSI colour model which gives the fire area in the image as output. The next step is fire area segmentation where the fire areas are roughly segmented and spurious fire areas like fire shadows and fire-like objects are removed by image difference. After this the change in shape of fire is estimated by taking contour image difference and white pixel ratio to estimate the burning degree of fire, i.e. no-fire, small, medium and large. The experimental results show that the method is able to detect different fire scenarios with relatively good accuracy.

2.2 Hybrid models for Image classification

The classifier part in a Deep CNN is a simple fully connected perceptron with a softmax layer at the end to output probabilities for each class. This section of the CNN has a high scope for improvement. Since it consists of three to four fully connected layers containing thousands of neurons, it becomes harder and slower to train it. Even with pre-trained models that require fine tuning of these layers. This has led to the development of hybrid CNN models, which consist of a specialist classifier at the end.
Some of the researchers have employed the Support Vector Machine (SVM) as the final stage classifier [1, 21, 25, 33, 38]. In [25], the CNN-SVM hybrid model is applied to many different problems like object classification, scene classification, bird sub-categorization, flower recognition etc. A linear SVM is fed 'off the shelf convolutional features' from the last layer of the CNN. This paper uses the OverFeat network [30] which is a state-of-the-art object classification model. The paper shows, with exhaustive experimentation, that extraction of convolutional features by a deep CNN is the best way to obtain relevant characteristics that distinguishes an entity from another.
The CNN-SVM model is used in [21] and successfully applied to visual learning and recognition for multi-robot systems and problems like human-swarm interaction and gesture recognition. This hybrid model has also been applied to gender recognition in [38]. The CNN used here is the AlexNet [17] pre-trained with ImageNet weights. The features extracted from the entire AlexNet are fed to an SVM classifier. A similar kind of research is done in [33], where the softmax layer and the cross-entropy loss are replaced by a linear SVM and margin loss. This model is tested on some of the most well known benchmark datasets like CIFAR-10, MNIST and Facial Expression Recognition challenge. The results show that this model outperforms the conventional Deep CNNs.
In 2006, G.B. Huang introduced a new learning algorithm for a single hidden layer feedforward neural network called the Extreme Learning Machine [13, 14]. This technique was many times faster than backpropagation and SVM, and

outperformed them on various tasks. The ELM randomly initializes the input weights and analytically determines the output weights. It produces a minimum norm least squares solution which always achieves lowest training accuracy, if there are enough number of hidden neurons. There have been many variants of ELM depending upon a specific application, which have been summarised in [12]. This led to the advent of CNN-ELM hybrid models, which were able to outperform the CNN-SVM models on various applications. The major advantage of CNN-ELM models is the speed of convergence. In [29], the CNN-ELM model is used for Wireless Capsule Endoscopy (WCE) image classification. The softmax classifier of a CNN is replaced by an ELM classifier and trained on the feature extracted by the CNN feature extractor. This model is able to outperform CNN-based classifiers.

The CNN-ELM model has also been used for handwritten digit classification [19, 22]. In [19], a 'shallow' CNN is used for feature extraction and ELM for classification. The shallow CNN together with ELM speeds up the training process. Also, various weight initialization strategies have been tested for ELM with different receptive fields. Finally, two strategies, namely the Constrained ELM (C-ELM) [44] and Computed Input Weights ELM (CIW-ELM) [35] are combined in a two layer ELM structure with receptive fields. This model was tested on the MNIST dataset and achieved 0.83% testing error. In [22], a deep CNN is used for the same application and tested on the USPS dataset.

A shallow CNN with ELM is tested on some benchmark datasets like MNIST, NORB-small, CIFAR-10 and SVHN with various hyper parameter configurations in [20]. Another similar hybrid model that uses CNN features and Kernel ELM as classifier is used in [9] for age estimation using facial features. Another application where a CNN-ELM hybrid model has been applied is the traffic sign recognition [41].

A different strategy of combining CNN feature extraction and ELM learning is proposed in [15]. Here, an ELM with single hidden layer is inserted after every convolution and pooling layer and at the end as classifier. The ELM is trained by borrowing values from the next convolutional layer and each ELM is updated after every iteration using backpropagation. This interesting architecture is applied to the application of lane detection and achieves excellent performance.

A comparative analysis of the CNN-ELM and CNN-SVM hybrid models for object recognition from ImageNet has been illustrated in [42]. Both these models were tested for object recognition from different sources like Amazon, Webcam, Caltech and DSLR. The final results show that the CNN-ELM model outperforms the CNN-SVM model on all datasets and using Kernel ELM further increases accuracy.

Using ELM as a final stage classifier does not end at image classification with CNNs. They have also been used with DBNs for various applications [3, 26].

3 The Fire Detector

In this paper, we propose to employ hybrid deep CNN models to perform fire detection. The AlexNet has been used by researchers in the past for fire detection which has produced satisfactory results. We propose to use two Deep CNN architectures that have outperformed the AlexNet on the ImageNet dataset, namely VGG16 [32] and Resnet50 [10]. We use these models with pre-trained ImageNet weights. This helps greatly when there is lack of training data. So, we fine-tune the ELM classifier on our dataset, which is fed the features extracted by the Deep CNNs.

3.1 Deep ConvNet Models

The Convolutional Neural Network was first introduced in 1980 by Kunihiko Fukushima [8]. The CNN is designed to take advantage of two dimensional structures like 2D Images and capture local spatial patterns. This is achieved with local connections and tied weights. It consists of one or more convolution layers with pooling layers between them, followed by one or more fully connected layers, as in a standard multilayer perceptron. CNNs are easier to train compared to Deep Neural Networks because they have fewer parameters and local receptive fields.
In CNNs, kernels/filters are used to see where particular features are present in an image by convolution with the image. The size of the filters gives rise to locally connected structure which are each convolved with the image to produce feature maps. The feature maps are usually sub-sampled using mean or max pooling. The reduction in parameters is due to the fact that convolution layers share weights.
The reason behind parameter sharing is that we make an assumption, that the statistics of a patch of a natural image are the same as any other patch of the image. This suggests that features learned at one location can also be learned for other locations. So, we can apply this learned feature detector anywhere in the image. This makes CNNs ideal feature extractors for images.
The CNNs with many layers have been used for various applications especially image classification. In this paper, we use two state-of-the-art Deep CNNs that have achieved one of the lowest error rates in image classification tasks.
In this work, we use VGG16 and Resnet50, pre-trained on the ImageNet dataset, along with a few modifications. We also compare our modified and hybrid models with the original ones. The VGG16 architecture was proposed by the Visual Geometry Group at the University of Oxford [32], which was deep, simple, sequential network whereas the Resnet50, proposed by Microsoft research [10], was an extremely deep graphical network with residual connections (which avoids the vanishing gradients problem and residual functions are easier to train).
We also test slightly modified versions of both these networks by adding a fully-connected layer and fine-tuning on our dataset. We also tested with more fully connected layers but the increase in accuracy was overshadowed by the increase in training time.

3.2 The Hybrid Model

We propose to use a hybrid architecture for fire detection in images. In this paper, instead of using a simple CNN as feature extractor, we employ state-of-the-art Deep CNNs like the VGG16 and Resnet50.
Figure 3(a) and 3(b) show the architecture of the VGG16-ELM and Resnet50-ELM hybrid models respectively. Usually, only the softmax classifier is replaced by another classifier (ELM or SVM) in a CNN to create a hybrid model. But, we go one step further by replacing the entire fully connected multi-layer perceptron with a single hidden layer ELM. This decreases the complexity of the model even further.

The Theory of Extreme Learning Machine: The Extreme Learning Machine is a supervised learning algorithm [13]. The input to the ELM, in this case, are the features extracted by the CNNs. Let it be represented as x_i, t_i, where x_i is the input feature instance and t_i is the corresponding class of the image. The inputs are connected to the hidden layer by randomly assigned weights w. The product of the inputs and their corresponding weights act as inputs to the hidden layer activation function. The hidden layer activation function is a non-linear non-constant bounded continuous infinitely differentiable function that maps the input data to the feature space. There is a catalogue of activation functions from which we can choose according to the problem at hand. We ran experiments for all activation functions and the best performance was achieved with the multiquadratics function:

$$f(x) = \sqrt{\|x_i - \mu_i\|^2 + a^2} \tag{1}$$

The hidden layer and the output layer are connected via weights β, which are to be analytically determined. The mapping from the feature space to the output space is linear. Now, with the inputs, hidden neurons, their activation functions, the weights connecting the inputs to the hidden layer and the output weights produce the final output function:

$$\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) = o_j \tag{2}$$

The output in Matrix form is:

$$H\beta = T \tag{3}$$

The error function used in Extreme Learning Machine is the Mean Squared error function, written as:

$$E = \sum_{j=1}^{N} (\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) - t_j)^2 \tag{4}$$

To minimize the error, we need to get the least-squares solution of the above linear system.

$$\|H\beta^* - T\| = min_\beta \|H\beta - T\| \tag{5}$$

The minimum norm least-squares solution to the above linear system is given by:

$$\hat{\beta} = H^{\dagger}T \tag{6}$$

Properties of the above solution:

1. *Minimum Training Error:* The following equation provides the least-squares solution, which means the solution for $\|H\beta - T\|$, i.e. the error is minimum.
   $\|H\beta^{*} - T\| = min_{\beta}\|H\beta - T\|$
2. *Smallest Norm of Weights:* The minimum norm of least-squares solution is given by the Moore-Penrose pseudo inverse of H.
   $\hat{\beta} = H^{\dagger}T$
3. *Unique Solution:* The minimum norm least-squares solution of $H\beta = T$ is unique, which is:
   $\hat{\beta} = H^{\dagger}T$

Detailed mathematical proofs of these properties and the ELM algorithm can be found in [14]. Both the VGG16 and Resnet50 extract rich features from the images. These features are fed to the ELM classifier which finds the minimum norm least squares solution. With enough number of hidden neurons, the ELM outperforms the original VGG16 and Resnet50 networks. Both VGG16 and Resnet50 are pre-trained with ImageNet weights. So, only the ELM classifier is trained on the features extracted by the CNNs.
Apart from fast training and accurate classification, there is another advantage of this model. This hybrid model does not require large training data. In fact, our dataset consists of just 651 images, out of which the ELM is trained on 60% of images only. This shows its robustness towards lack of training data. A normal Deep CNN would require much higher amount of training data to fine-tune its fully-connected layers and the softmax classifier. Even the pre-trained VGG16 and Resnet50 models required at least 80% training data to fine-tune their fully-connected layers.
And, as we will show in the next section, a hybrid CNN-ELM trained with 60% training data outperforms pre-trained VGG16 and Resnet50, fine-tuned on 80% training data.

3.3 Paper Contributions

1. The previous hybrid models have used simple CNNs for feature extraction. We employ state-of-the-art Deep CNNs to make feature extraction more efficient and obtain relevant features since the dataset is difficult to classify.
2. Other hybrid models simply replace the softmax classifier with SVM or sometimes ELM. We completely remove the fully connected layers to increase speed of convergence since no fine-tuning is needed and also reduce the complexity of the architecture. Since VGG16 and Resnet50 extract rich features and the ELM is an accurate classifier, we do not need the fully-connected layers. This decreases the number of layers by 2 in VGG16 and by 1 in Resnet50, which is 8192 and 4096 neurons respectively.

3. The above point also justifies the use of complex features extractors like VGG16 and Resnet50. If we used a simple CNN then, we might not be able to remove the fully-connected layers since the features might not be rich enough. Due to this, the fully-connected layers would have to be fine-tuned on the dataset which would increase training time and network complexity.
4. Also, we see that the data required for training the ELM classifier is lower than the data required for fine-tuning the fully-connected layers of a pre-trained Deep CNN.
5. We apply our hybrid model on the problem of fire detection in images (on our own dataset). And, to the best of our knowledge, this is the first time a hybrid ELM model has been applied to this problem.

4 Experiments

We conducted our experiments to compare training and testing accuracies and execution times of: the VGG16 and Resnet50 models including modifications, Hybrid VGG16 and Resnet50 models with ELM classifier. We also compare our hybrid VGG16-ELM and Resnet50-ELM models with VGG16-SVM and Resnet50-SVM as well. We used pre-trained Keras [5] models and fine-tune the fully-connected layers on our dataset. The training of the models was done on the following hardware specifications: Intel i5 2.5GHz, 8GB RAM and Nvidia Geforce GTX 820 2GB GPU. Each model was trained on the dataset for 10 training epochs. The ADAM optimizer [16] with default parameters $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ was used to fine-tune the fully-connected layers for VGG16 and Resnet50 and their modified versions. The details of the dataset are given in the next subsection.

4.1 The Real World Fire Dataset

Since there is no benchmark dataset for fire detection in images, we created our own dataset by handpicking images from the internet. [2]This dataset consists of 651 images which is quite small in size but it enables us to test the generalization capabilities and the effectiveness and efficiency of models to extract relevant features from images when training data is scarce. The dataset is divided into training and testing sets. The training set consists of 549 images: 59 fire images and 490 non-fire images. The imbalance is deliberate to replicate real world situations, as the probability of occurrence of fire hazards is quite small. The datasets used in previous papers have been balanced which does not imitate the real world environment. The testing set contains 102 images: 51 images each of fire and non-fire classes. As the training set is highly unbalanced and the testing set is exactly balanced, it makes a good test to see whether the models are able to generalize well or not. For a model with good accuracy, it must be able to

[2] The dataset is available here: `https://github.com/UIA-CAIR/Fire-Detection-Image-Dataset`

extract the distinguishing features from the small amount of fire images. To extract such features from small amount of data the model must be deep enough. A poor model would just label all images as non-fire, which is exemplified in the results.
Apart from being unbalanced, there are a few images that are very hard to classify. The dataset contains images from all scenarios like fire in a house, room, office, forest fire, with different illumination intensity and different shades of red, yellow and orange, small and big fires, fire at night, fire in the morning. Non-fire images contain a few images that are hard to distinguish from fire images like a bright red room with high illumination, sunset, red coloured houses and vehicles, bright lights with different shades of yellow and red etc.
The figures 4(a) to 4(f) show fire images in different environments: indoor, outdoor, daytime, nighttime, forest fire, big and small fire. And the figures 5(a) to 5(f) show the non-fire images that are difficult to classify. Considering these characteristics of our dataset, detecting fire can be a difficult task. We have made the dataset available online so that it can be used for future research in this area.

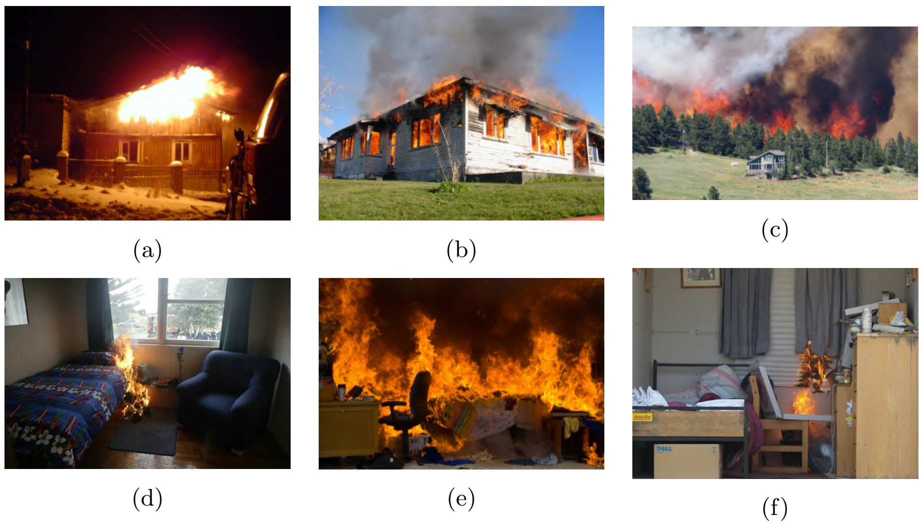


Fig. 1: Examples of Fire Images

4.2 Results

Our ELM hybrid models are tested on our dataset and compared with SVM hybrid models and the original VGG16 and Resnet50 Deep CNN models. Table 1 and Table 2 show the results of the experiments. The dataset was randomly split into training and testing sets. Two cases were considered depending on the amount of training data. The Deep CNN models (VGG16 and Resnet50) were

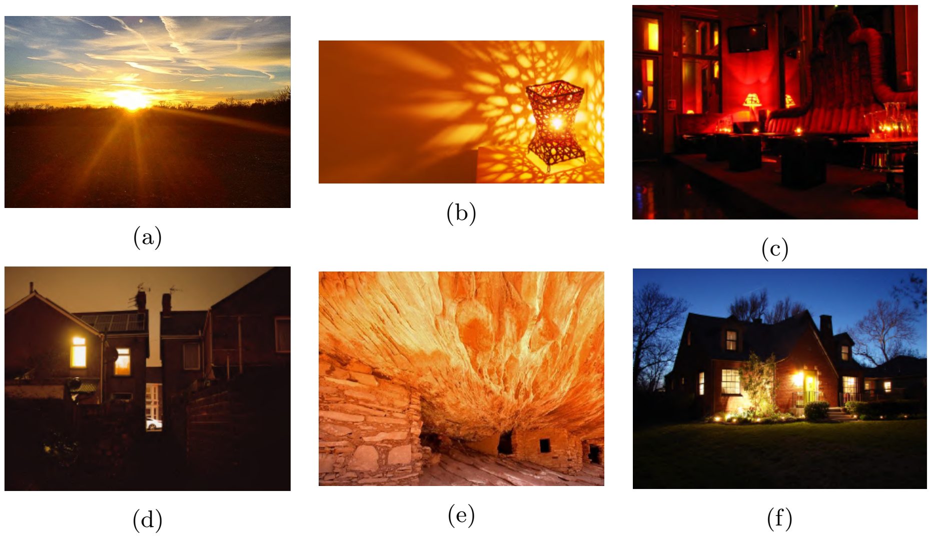


Fig. 2: Examples of Non-Fire Images that are difficult to classify

Table 1: Accuracy and Execution time

Model	D_T	Acc_{train}	T_{train}	T^C_{train}	Acc_{test}	T_{test}
VGG16 (pre-trained)	80	100	7149	6089	90.19	121
VGG16 (modified)	80	100	7320	6260	91.176	122
Resnet50 (pre-trained)	80	100	15995	13916	91.176	105
Resnet50 (modified)	80	100	16098	13919	92.15	107
VGG16+SVM	60	99.6	2411	1352	87.4	89
VGG16+SVM	80	100	2843	1784	93.9	81
VGG16+ELM	60	100	1340	281	93.9	24
VGG16+ELM	80	100	1356	297	96.15	21
Resnet50+SVM	60	100	3524	1345	88.7	97
Resnet50+SVM	80	100	4039	1860	94.6	86
Resnet50+ELM	60	100	2430	251	98.9	32
Resnet50+ELM	80	100	2452	272	99.2	26

D_T is the percentage of total data used for training the models.
Acc_{train} and Acc_{test} are the training and testing accuracies respectively.
T_{train} and T_{test} are the training and testing times for the models.
T^C_{train} is the time required to train the classifier part of the models.

trained only on 80% training data, since 60% is too less for these models. All the hybrid models have been trained on both 60% and 80% of training data. One point to be noted here is that, the SVM hybrid models contain an additional fully-connected layer of 4096 neurons, while the ELM is directly connected to the last pooling layer.

Table 2: Number of Hidden Neurons in ELM

CNN Features	**# hidden neurons**	**Testing accuracy**
VGG16 Feature Extractor	4096	93.9
VGG16 Feature Extractor	8192	94.2
VGG16 Feature Extractor	16384	91.1 (Overfitting)
Resnet50 Feature Extractor	4096	98.9
Resnet50 Feature Extractor	8192	99.2
Resnet50 Feature Extractor	16384	96.9 (Overfitting)

The results in Table 1 show that the ELM hybrid models outperform the VGG16, Resnet50 and SVM hybrid models by achieving higher accuracy and learning much faster. In general, we can see that the hybrid models outperform the state-of-the-art Deep CNNs in terms of both accuracy and training time.
Apart from accuracy and training time, another important point drawn from the results is the amount of training data required. As we already know, Deep Neural Networks (DNN) require huge amount of training data. So, using pre-trained models can be highly beneficial, as we only need to fine-tune the fully-connected layers. But, with models like VGG16 and Resnet50 which have large fully-connected layers, even fine-tuning requires large amount of training data. We had to train the VGG16 and Resnet50 on at least 80% training data otherwise they were overfitting on the majority class, resulting in 50% accuracy.
But in case of hybrid models, especially ELM hybrid models, the amount of training data required is much less. Even after being trained on 60% training data, the ELM models were able to outperform the original VGG16 and Resnet50 models which were trained on 80% training data. This shows that reducing the fully-connected layers, or replacing them with a better classifier can reduce the amount of training data required. Also, the ELM is more robust towards lack of training data which adds to this advantage.
Among the hybrid models, the ELM hybrid models outperform the SVM hybrid models both in terms of testing accuracy and training time. Also, we can see that the hybrid models with Resnet50 as the feature extractor achieves better results than the hybrid models with VGG16 as the feature extractor. This is due to the depth and the residual connections in Resnet50 in contrast to the simple, shallower (compared to Resnet50) and sequential nature of VGG16.
Table 2 compares results between different number of hidden neurons used by ELM. The accuracy increases as the number of hidden neurons increase. The models are tested for 2^{12}, 2^{13} and 2^{14} number of neurons. The testing accuracy starts to decrease for 2^{14} neurons, which means the model overfits. All the tests in Table 2 were conducted with 60% training data.

5 Conclusion

In this paper, we have proposed a hybrid model for fire detection. The hybrid model combines the feature extraction capabilities of Deep CNNs and the classification ability of ELM. The Deep CNNs used for creating the hybrid models are the VGG16 and Resnet50 instead of a simple Deep CNN. The fully connected layers are removed completely and replaced by a single hidden layer feedforward neural network trained using the ELM algorithm. This decreases complexity of the network and increases speed of convergence. We test our model on our own dataset which has been created to replicate a realistic view of the environment which includes different scenarios, imbalance due to lower likelihood of occurrence of fire. The dataset is small in size to check the robustness of models towards lack of training data, since deep networks require a considerable amount of training data. Our hybrid model is compared with the original VGG16 and Resnet50 models and also with SVM hybrid models. Our Deep CNN-ELM model is able to outperform all other models in terms of accuracy by 2.8% to 7.1% and training time by a speed up of 20x to 51x and requires less training data to achieve higher accuracy for the problem of fire detection.

References

1. Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. *CoRR*, abs/1406.5774, 2014.
2. G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.
3. Le-le Cao, Wen-bing Huang, and Fu-chun Sun. *A Deep and Stable Extreme Learning Approach for Classification and Regression*, pages 141–150. Springer International Publishing, Cham, 2015.
4. Daniel Yoshinobu Takada Chino, Letricia P. S. Avalhais, José Fernando Rodrigues Jr., and Agma J. M. Traina. Bowfire: Detection of fire in still images by integrating pixel color and texture analysis. *CoRR*, abs/1506.03495, 2015.
5. Francois Chollet. Keras, 2015.
6. J. Zhao Z. Zhang C. Qu Y. Ke D. Zhang, S. Han and X. Chen. Image based forest fire detection using dynamic characteristics with artificial neural networks. In *2009 International Joint Conference on Artificial Intelligence*, pages 290–293, April 2009.
7. S. Frizzi, R. Kaabi, M. Bouchouicha, J. M. Ginoux, E. Moreau, and F. Fnaiech. Convolutional neural network for video fire and smoke detection. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 877–882, Oct 2016.
8. Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
9. F. Grpinar, H. Kaya, H. Dibeklioglu, and A. A. Salah. Kernel elm and cnn based facial age estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 785–791, June 2016.

10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
11. Wen-Bing Horng and Jian-Wen Peng. Image-based fire detection using neural networks. In *JCIS*, 2006.
12. Gao Huang, Guang-Bin Huang, Shiji Song, and Keyou You. Trends in extreme learning machines: a review. *Neural Networks*, 61:32–48, 2015.
13. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.
14. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.
15. Jihun Kim, Jonghong Kim, Gil-Jin Jang, and Minho Lee. Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection. *Neural Networks*, 87:109 – 121, 2017.
16. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
17. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
18. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
19. Mark D. McDonnell, Migel D. Tissera, André van Schaik, and Jonathan Tapson. Fast, simple and accurate handwritten digit classification using extreme learning machines with shaped input-weights. *CoRR*, abs/1412.8307, 2014.
20. Mark D. McDonnell and Tony Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. *CoRR*, abs/1503.04596, 2015.
21. Jawad Nagi, Gianni A. Di Caro, Alessandro Giusti, Farrukh Nagi, and Luca Maria Gambardella. Convolutional neural support vector machines: Hybrid visual pattern classifiers for multi-robot systems. In *ICMLA (1)*, pages 27–32. IEEE, 2012.
22. Shan Pang and Xinyi Yang. Deep convolutional extreme learning machine and its application in handwritten digit classification. *Intell. Neuroscience*, 2016:6–, August 2016.
23. Bc. Tomas Polednik. Detection of fire in images and video using cnn. *Excel@FIT*, 2015.
24. K. Poobalan and S.C. Liew. Fire detection algorithm using image processing techniques. In *3rd International Conference on Artificial Intelligence and Computer Science (AICS2015)*, Ocotober 2015.
25. Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.
26. Bernardete Ribeiro and Noel Lopes. *Extreme Learning Classifier with Deep Concepts*, pages 182–189. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
27. Richard Bright Richard Custer. Fire detection: The state of the art. *NBS Technical Note, US Department of Commerce*, 1974.
28. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

B

29. J. s. Yu, J. Chen, Z. Q. Xiang, and Y. X. Zou. A hybrid convolutional neural networks with extreme learning machine for wce image classification. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1822–1827, Dec 2015.
30. Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.
31. Jing Shao, Guanxiang Wang, and Wei Guo. An image-based fire detection method using color analysis. In *2012 International Conference on Computer Science and Information Processing (CSIP)*, pages 1008–1011, Aug 2012.
32. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
33. Yichuan Tang. Deep learning using support vector machines. *CoRR*, abs/1306.0239, 2013.
34. C. Tao, J. Zhang, and P. Wang. Smoke detection based on deep convolutional neural networks. In *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, pages 150–153, Dec 2016.
35. Jonathan Tapson, Philip de Chazal, and André van Schaik. Explicit computation of input weights in extreme learning machines. *CoRR*, abs/1406.2889, 2014.
36. Tom Toulouse, Lucile Rossi, Turgay Celik, and Moulay Akhloufi. Automatic fire pixel detection using image processing: a comparative analysis of rule-based and machine learning-based methods. *Signal, Image and Video Processing*, 10(4):647–654, 2016.
37. B. Ugur Treyin, Yigithan Dedeoglu, Ugur Gdkbay, and A. Enis etin. Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters*, 27(1):49 – 58, 2006.
38. J. v. d. Wolfshaar, M. F. Karaaba, and M. A. Wiering. Deep convolutional neural networks and support vector machines for gender recognition. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 188–195, Dec 2015.
39. Steven Verstockt, Peter Lambert, Rik Van de Walle, Bart Merci, and Bart Sette. State of the art in vision-based fire and smoke dectection. In Heinz Luck and Ingolf Willms, editors, *International Conference on Automatic Fire Detection, 14th, Proceedings*, volume 2, pages 285–292. University of Duisburg-Essen. Department of Communication Systems, 2009.
40. Jerome Vicente and Philippe Guillemant. An image processing technique for automatically detecting forest fire. *International Journal of Thermal Sciences*, 41(12):1113 – 1120, 2002.
41. Yujun Zeng, Xin Xu, Yuqiang Fang, and Kun Zhao. *Traffic Sign Recognition Using Deep Convolutional Networks and Extreme Learning Machine*, pages 272–280. Springer International Publishing, Cham, 2015.
42. Lei Zhang and David Zhang. SVM and ELM: who wins? object recognition with deep convolutional features from imagenet. *CoRR*, abs/1506.02509, 2015.
43. Qingjie Zhang, Jiaolong Xu, Liang Xu, and Haifeng Guo. Deep convolutional neural networks for forest fire detection. February 2016.
44. W. Zhu, J. Miao, and L. Qing. Constrained extreme learning machine: A novel highly discriminative random feedforward neural network. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 800–807, July 2014.

Paper C

Title:	Deep Q-Learning with Q-Matrix Transfer Learning for Novel Fire Evacuation Environment
Authors:	Jivitesh Sharma, Per-Arne Andersen, Ole-Christoffer Granmo and Morten Goodwin
Affiliation:	Dept. of Information and Communication Technology, University of Agder (UiA), Grimstad, Norway
Journal:	*IEEE Transactions on Systems, Man and Cybernetics*, pages 1-19., DOI: 10.1109/TSMC.2020.2967936, Feb. 2020.
Copyright ©:	IEEE

Deep Q-Learning with Q-Matrix Transfer Learning for Novel Fire Evacuation Environment

Jivitesh Sharma*, Per-Arne Andersen†, Ole-Christoffer Granmo‡ and Morten Goodwin§
Centre for Artificial Intelligence Research
Department of Information and Communication Technology
University of Agder, Norway
*jivitesh.sharma@uia.no, †per@sysx.no, ‡ole.granmo@uia.no, §morten.goodwin@uia.no

Abstract—**Deep Reinforcement Learning is achieving significant success in various applications like control, robotics, games, resource management, and scheduling. However, the important problem of emergency evacuation, which clearly could benefit from reinforcement learning, has been largely unaddressed. Indeed, emergency evacuation is a complex task which is difficult to solve with reinforcement learning. An emergency situation is highly dynamic, with a lot of changing variables and complex constraints that make it challenging to solve. Also, there is no standard benchmark environment available that can be used to train Reinforcement Learning agents for evacuation. A realistic environment can be complex to design. In this paper, we propose the first fire evacuation environment to train reinforcement learning agents for evacuation planning. The environment is modelled as a graph capturing the building structure. It consists of realistic features like fire spread, uncertainty and bottlenecks. We have implemented the environment in the OpenAI gym format, to facilitate future research. We also propose a new reinforcement learning approach that entails pretraining the network weights of a DQN based agent (DQN/Double-DQN/Dueling-DQN) to incorporate information on the shortest path to the exit. We achieved this by using tabular Q-learning to learn the shortest path on the building model's graph. This information is transferred to the network by deliberately overfitting it on the Q-matrix. Then, the pretrained DQN model is trained on the fire evacuation environment to generate the optimal evacuation path under time varying conditions due to fire spread, bottlenecks and uncertainty. We perform comparisons of the proposed approach with state-of-the-art reinforcement learning algorithms like DQN, DDQN, Dueling-DQN, PPO, VPG, SARSA, A2C and ACKTR. The results show that our method is able to outperform state-of-the-art models by a huge margin including the original DQN based models. Finally, we test our model on a large and complex real building consisting of 91 rooms, with the possibility to move to any other room, hence giving 8281 actions. In order to reduce the action space, we propose a strategy that involves one step simulation. That is, an action importance vector is added to the final output of the pretrained DQN and acts like an attention mechanism. Using this strategy, the action space is reduced by 90.1%. In this manner, we are able to deal with large action spaces. Hence, our model achieves near optimal performance on the real world emergency environment.**

Index Terms—**Reinforcement Learning, Deep Q-Networks, DQN, Double DQN, Dueling DQN, Pretraining, Transfer Learning, Fire Evacuation Environment, Emergency Management, Evacuation.**

I. Introduction

REINFORCEMENT Learning (RL) has been a subject of extensive research and applications in various real world domains such as Robotics, Games, Industrial Automation and Control, System Optimization, Quality Control and Maintenance. But, some extremely important areas, where Reinforcement Learning could be immensely vital, have not received adequate attention from researchers. We turn our attention to the major problem of evacuation in case of fire emergencies.

Fire related disasters are the most common type of Emergency situation. They require thorough analysis of the situation for quick and precise response. Even though this critical application hasn't received adequate attention from AI researchers, there have been some noteworthy contributions. One such paper, focusing on assisting decision making for fire brigades, is described in [1]. Here, the the RoboCup Rescue simulation is used as a fire simulation environment [2]. A SARSA Agent [3] is used with a new learning strategy called Lesson-by-Lesson learning, similar to curriculum learning. Results show that the RL agent is able to perform admirably in the simulator. However, the simulator lacks realistic features like bottlenecks, fire spread and has a grid structure which is too simplistic to model realistic environments. Also, the approach seems unstable and needs information about the state which isn't readily available in real life scenarios.

In [4], multiple coordinated agents are used for forest fire fighting. The paper uses a software platform called Pyrosim which is used to create dynamic forest fire situations. The simulator is mostly used for terrain modeling and a coordinated multiple agent system is used to extinguish fire and not for evacuation.

The evacuation approach described in [5] is similar to the problem we try to solve in this paper. In [5], a fading memory mechanism is proposed with the intuition that in dynamic environments less trust should be put on older knowledge for decision making. But arguably, this could be achieved more efficiently by the 'γ' parameter in Q-learning along with prioritized experience replay. Also, the graph based environment used in [5] lacks many key features like fire spread, people in rooms, bottlenecks etc.

The most significant work done on building evacuation using RL is reported in [6]. The evacuation environment is grid based with multiple rooms and fire. The fire spread is modelled accurately and uncertainty taken into account. The multi-agent Q-learning model is shown to work in large spaces as well. Further, the paper demonstrates a simple environment and strategy for evacuation. However, the approach proposed in [6] lacks key features like bottlenecks and actual people in rooms. The grid based environment isn't able to capture details of the building model like room locations and paths connecting rooms.

Some interesting research on evacuation planning take a completely different approach by simulating and modelling human and crowd behaviour under evacuation [7]–[10]. Our work on evacuation planning is not based on human behaviour modelling or the BDI (Belief-Desire-Intention) framework for emergency scenarios. These methods are beyond the scope of this paper and not discussed here.

Proposed Environment: There are many reinforcement learning libraries that contain simulations and game environments to train reinforcement learning based agents [11]–[15]. However, currently no realistic learning environment for emergency evacuation has been reported.

In our paper, we build the first realistic fire evacuation environment specifically designed to train reinforcement learning agents for evacuating people in the safest manner in the least number of time-steps possible. The environment has the same structure as OpenAI gym environments, so it can be used easily in the same manner.

The proposed fire evacuation environment is graph based, which requires complex decision making such as routing, scheduling and dealing with bottlenecks, crowd behaviour uncertainty and fire spread. This problem falls in the domain of discrete control. The evacuation is performed inside a building model, which is represented as a graph. The agent needs to evacuate all persons in all rooms through any available exits using the shortest path in the least number of time-steps, while avoiding any perilous situations like the fire, bottlenecks and other hazardous situations.

Some previous research papers focus on modelling fire spread and prediction, mostly using cellular automata [16] and other novel AI techniques [17]–[19]. An effective and innovative way of modelling fire spread is to use spatial reinforcement learning, as proposed in [20]. However, our way of simulating fire spread is far less complex and leverages rewarding system of the RL framework. In our proposed environment, we simply use an exponential decay reward function to model the fire spread and direction. To keep in tune with the RL framework, the feedback from the environment sent back to the agent should convey enough information. So, we design the reward function in such a manner that the agent can learn about the fire spread and take measures accordingly.

Proposed Method: Since this environment poses a high level of difficulty, we argue that incorporating the shortest path information (shortest path from each room to the nearest exit) in the DQN model(s) by transfer learning and pretraining the DQN neural network function approximator is necessary.

Transfer learning has been used extensively in computer vision tasks for many years, recently vastly expanded for many computer vision problems in [21]. Lately, it has been utilized in Natural Language models [22], [23]. In reinforcement learning, pretrained models have started to appear as well [24], [25]. In fact, we use the convergence analysis of [25], which provides a general theoretical perspective of task transfer learning, to prove that our method guarantees convergence.

In this paper, we present a new class of pretrained DQN models called Q-matrix Pretrained Deep Q-Networks (QMP-DQN). We employ Q-learning to learn a Q-matrix representing the shortest paths from each room to the exit. We perform multiple random episodic starts and ϵ-greedy exploration of the building model graph environment. Q-learning is applied on a pretraining instance of the environment that consists of only the building model graph. Then, we transfer the Q-matrix to a DQN model, by pretraining the DQN to reproduce the Q-matrix. Finally, we train the pretrained DQN agent on the complete fire evacuation task. We compare our proposed pretrained DQN models (QMP-DQN) against regular DQN models and show that pretraining for our fire evacuation environment is necessary. We also compare our QMP-DQN models with state-of-the-art Reinforcement Learning algorithms and show that off-policy Q-learning techniques perform better than other policy based methods as well as actor-critic models.

Finally, in Section 5, we show that our method can perform optimal evacuation planning on a large and complex real world building model by dealing with the large discrete action space in a new and simple way by using an attention based mechanism.

Contributions: This paper contributes to the field of reinforcement learning, emergency evacuation and management in the following manner:

1) We propose the first reinforcement learning based fire evacuation environment with OpenAI Gym structure.
2) We build a graph based environment to accurately model the building structure, which is more efficient than a maze structure.
3) The environment can consist of a large discrete action space with n^2 number of actions (for all possibilities), where n is the number of rooms in the building. That is, the action space size increases exponentially with respect to the rooms.
4) Our proposed environment contains realistic features such as multiple fires and dynamic fire spread which is modelled by the exponential decay reward function.
5) We further improve the realism of our environment by restricting the number of people allowed in each room to model over-crowded hazardous situations.
6) We also include uncertainty about action performed in the environment to model uncertain crowd behaviour, which also acts as a method of regularization.
7) We use the Q-matrix to transfer learned knowledge of the shortest path by pretraining a DQN agent to reproduce the Q-matrix.
8) We also introduce a small amount of noise in the Q-matrix, to avoid stagnation of the DQN agent in a local optimum.
9) We perform exhaustive comparisons with other state-of-the-art reinforcement learning algorithms like DQN, DDQN, Dueling DQN, VPG, PPO, SARSA, A2C and ACKTR.

10) We test our model on a large and complex real world scenario, which is the University of Agder Building, which consists of 91 nodes, and 8281 actions.
11) We propose a new and simple way to deal with large discrete action spaces in our proposed environment, by employing an attention mechanism based technique.

The rest of the paper is organized as follows: Section 2 summarizes the RL concepts used in this paper. Section 3 gives a detailed explanation of the proposed Fire Emergency Evacuation System with each module of the system described in subsequent sub-sections. Section 4 reports our exhaustive experimental results. Section 5 presents the real world application of our model in a large and complex environment, and finally Section 6 concludes the paper.

II. Preliminaries

Reinforcement Learning is a sub-field of Machine Learning which deals with learning to make appropriate decisions and take actions to achieve a goal. A Reinforcement Learning agent learns from direct interactions with an environment without requiring explicit supervision or a complete model of the environment. The agent interacts with the environment by performing actions. It receives feedback for it's actions in terms of reward (or penalty) from the environment and observes changes in the environment as a result of the actions it performs. These observations are called states of the environment and the agent interacts with the environment at discrete time intervals t by performing an action a_t in a state of the environment s_t, it transitions to a new state s_{t+1} (change in the environment) while receiving a reward r_t, with probability $P(s_{t+1}|s_t, a_t)$. The main aim of the agent is to maximize the cumulative reward over time through it's choice of actions. A pictorial representation of the RL framework is shown in Fig. 1.
In the subsequent subsections, a brief presentation of the concepts and methods used in this paper are explained.

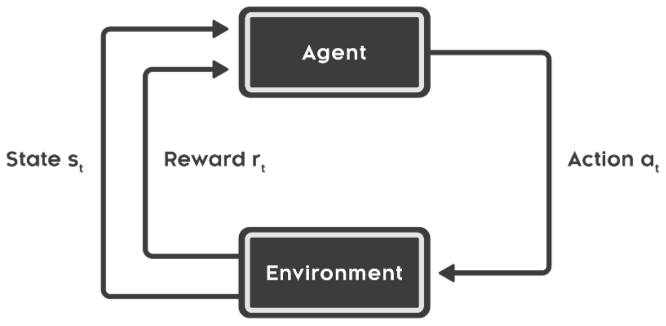


Fig. 1: Reinforcement Learning Framework (the figure is taken from [26])

A. *Markov Decision Process*

The Reinforcement learning framework is formalised by Markov Decision Processes (MDP) which are used to define the interaction between a learning agent and its environment in terms of states, actions, and rewards [27]. An MDP consists of a tuple of $\langle S, A, P, R\rangle$ [26], where S is the state space, A is the action space, P is the transition probability from one state to the next, $P : S \times A \times S \longmapsto [0, 1]$ and R is the reward function, $R : S \times A \longmapsto \mathbb{R}$.
When state space S, action space A and rewards R consist of finite number of elements, s_{t+1} and r_{t+1} have well-defined discrete probability distributions which depend only on the present state and action (Markov Property). This is represented as $p(s_{t+1}, r_{t+1}|s_t, a_t)$, where p determines the dynamics of the Markov Decision Process and where:

$$\sum_{s_{t+1}\in S}\sum_{r\in R} p(s_{t+1}, r_{t+1}|s_t, a_t) = 1, \forall s_t \in S, a_t \in A \quad (1)$$

p contains all the information about the MDP, so we can compute important aspects about the environment from p, like state transition probability and expected rewards for state-action pairs [26]:

$$P(s_{t+1}|s_t, a_t) = \sum_{r\in R} p(s_{t+1}, r_{t+1}|s_t, a_t) \quad (2)$$

$$r(s_t, a_t) = \mathbb{E}[r_t|s_t, a_t] = \sum_{r\in R} r \sum_{s_{t+1}\in S} p(s_{t+1}, r_{t+1}|s_t, a_t) \quad (3)$$

The equation 3, gives the immediate reward we expect to get when performing action a_t from state s_t. The agent tries to select actions that maximize the sum of rewards it expects to achieve, as time goes to infinity. But, in a dynamic and/or continuous Markov Decision Process, the notion of discounted rewards is used [26]:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (4)$$

where, γ is the discount factor and is in the range $[0, 1]$. If γ is near 0, then the agent puts emphasis on rewards received in the near future and if γ is near 1, then the agent also cares about rewards in the distant future.
In order to maximize G_t, the agent picks an action a_t when in a state s_t according to a policy function $\pi(s_t)$. A policy function is a probabilistic mapping from the state space to the action space, $S \rightarrow A$. The policy function outputs probabilities for taking each action in give state, so it can also be denoted as $\pi(a_t|s_t)$.

B. *Q-Learning*

Most of the Reinforcement Learning algorithms (value based) try to estimate the value function which gives an estimate of how good a state is for the agent to reside in. This is estimated according to the expected reward of a state under a policy and is denoted as $v_\pi(s)$:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|s_t] \quad (5)$$

Q-learning is a value based Reinforcement Learning algorithm that tries to maximize the q function [28]. The q function is a state-action value function and is denoted by $Q(s_t, a_t)$. It tries to maximize the expected reward give a state and action performed on that state:

$$Q(s_t, a_t) = \mathbb{E}[G_t|s_t, a_t] \quad (6)$$

C

According the Bellman Optimality equation [26], the optimal q function can be obtained by:

$$\begin{aligned} Q^*(s_t, a_t) &= \mathbb{E}[r_{t+1} + \gamma v^*(s_{t+1}) | s_t, a_t] \\ &= \sum_{s_{t+1}, r_t} p(s_{t+1}, r_t | s_t, a_t)[r_t + \gamma v^*(s)] \end{aligned} \tag{7}$$

where, $v^*(s_{t+1}) = \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$. And, a^* is the optimal action which results in maximum reward, the optimal policy is formed as $\arg\max_{a_t} \pi^*(a_t|s_t) = a^*$. This method was proposed in [28] which is tabular style Q-learning. The update rule for each time step of Q-learning is as follows:

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \eta[r_t + \gamma \max_{a_t} Q_t(s_{t+1}, a_{t+1}) \\ - Q_t(s_t, a_t)] \end{aligned} \tag{8}$$

Q-learning is an incremental dynamic programming algorithm that determines the optimal policy in a step-by-step manner. At each step t, the agent performs the following operations:

- Observes current state s_t.
- Selects and performs an action a_t.
- Observes the next state s_{t+1}.
- Receives the reward r_t.
- Updates the q-values $Q_t(s_t, a_t)$ using equation 8.

The q value function converges to the optimal value $Q_{t+1}(s_t, a_t) \rightarrow Q^*(s_t, a_t)$ as $t \rightarrow \infty$. Detailed convergence proof and analysis can be found in [28].
This tabular Q-learning method is used in our proposed approach to generate a Q-matrix for the shortest path to the exit based on the building model. In order to incorporate the shortest path information, this Q-matrix is used to pretrain the DQN models.

C. *Deep Q Network*

The tabular Q-learning approach works well for small environments, but becomes infeasible for complex environments with large multidimensional discrete or continuous state-action spaces. To deal with this problem, a parameterized version of the q function is used for approximation $Q(s_t, a_t; \theta) \approx Q^*(s_t, a_t)$. This way of function approximation was first proposed in [29].
Deep Neural Networks (DNNs) have become the predominant method for approximating complex intractable functions. They have become the defacto method for various applications such as image processing and classification [30]–[36], speech recognition [37]–[43], and natural language processing [44]–[48]. DNNs have also been applied to reinforcement learning problems successfully by achieving noteworthy performance [49], [50].
The most noteworthy research in integrating deep neural networks and Q-learning in an end-to-end reinforcement learning fashion is the Deep Q-Networks (DQNs) [51], [52]. To deal with the curse of dimensionality, a neural network is used to approximate the parameterised Q-function $Q(s_t, a_t; \theta)$. The neural network takes a state as input and approximates Q-values for each action based on the input state. The parameters are updated and the Q-function is refined in every iteration through an appropriate optimizer like Stochastic Gradient Descent [53], RMSProp [54], Adagrad [55], Adam [56] etc. The neural network outputs q-values for each action for the input state and the action with the highest q-value is selected (There is another DQN architecture, which is less frequently used, that takes in the state and action as input and returns it's q-value as output).
The DQN can be trained by optimizing the following loss function:

$$L_i(\theta_i) = \mathbb{E}[(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) - Q(s_t, a_t; \theta_i))^2] \tag{9}$$

where, γ is the discount factor, θ_i and θ_{i-1} are the Q-network parameters at iteration i and $i-1$ respectively. In order to train the Q-network, we require a target to calculate loss and optimize parameters. The target q-values are obtained by holding the parameters θ_{i-1} fixed from the previous iteration.

$$y = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) \tag{10}$$

where, y is the target for the next iteration to refine the Q-network. Unlike supervised learning where the optimal target values are known and fixed prior to learning, in DQN the approximate target values y, which depend on network parameters, are used to train the Q-network. The loss function can be rewritten as:

$$L_i(\theta_i) = \mathbb{E}[(y - Q(s_t, a_t; \theta_i))^2] \tag{11}$$

The process of optimizing the loss function $L_i(\theta_i)$ at the i^{th} iteration by holding the parameters from the previous iteration θ_{i-1} fixed, to get target values, results in a sequence of well-defined optimization time-steps. By differentiating the loss function in equation 11, we get the following gradient:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}[(y - Q(s_t, a_t; \theta_i)) \nabla_{\theta_i} Q(s_t, a_t; \theta_i)] \tag{12}$$

Instead of computing the full expectation of the above gradient, we optimize the loss function using an appropriate optimizer (in this paper we use the Adam optimizer [56]). The DQN is a model-free algorithm since it directly solves tasks without explicitly estimating the environment dynamics. Also, DQN is an off-policy method as it learns a greedy policy $a = \arg\max_{a_{t+1}} Q(s, a_{t+1}; \theta)$, while following an ϵ-greedy policy for sufficient exploration of the state space. One of the drawbacks of using a nonlinear function approximator like neural network is that it tends to diverge and is quite unstable for reinforcement learning. The problem of instability arises mostly due to: correlations between subsequent observations and that small changes in q-values can significantly change the policy and the correlations between q-values and target values.
The most well-known and simple technique to alleviate the problem of instability is the experience replay [57]. At each time-step, a tuple consisting of the agent's experience $E_t = (s_t, a_t, r_t, s_{t+1})$ is stored in a replay memory over many episodes. A minibatch of these tuples is randomly drawn from the replay memory to update the DQN parameters. This ensures that the network isn't trained on a sequence of observations (avoiding strong correlations between samples and reducing variance between updates) and it increases sample efficiency. This technique greatly increases stability of DQN.

D. Double DQN

Q-learning and DQN are capable of achieving performance beyond the human level on many occasions. However, in some cases Q-learning performs poorly and so does its deep neural network counterpart DQN. The main reason behind such poor performance is that Q-learning tends to overestimate action values. These overestimations are caused due to a positive bias that results from the $\max$ function in Q-learning and DQN updates which outputs the maximum action value as an approximation of the maximum expected action value.
The Double Q-learning method was proposed in [58] to alleviate this problem and later extended to DQN [59] to produce the Double DQN (DDQN) method. Since Q-learning uses the same estimator to select and evaluate an action, which results in overoptimistic action values, we can interpret it as a single estimator. In Double Q-learning, the task of evaluation and selection is decoupled by using double estimator approach consisting of two functions: Q^A and Q^B. The Q^A function is updated with a value from the Q^B function for the next state and the Q^B function is updated with a value from the Q^A function for the next state.
Let,

$$a^* = \arg\max_{a_t} Q_t^A(s_{t+1}, a_t) \tag{13}$$

$$b^* = \arg\max_{a_t} Q_t^B(s_{t+1}, a_t) \tag{14}$$

Then,

$$Q_{t+1}^A(s_t, a_t) = Q_t^A(s_t, a_t) + \eta[r_t + \gamma Q_t^B(s_{t+1}, a^*) - Q_t^A(s_t, a_t)] \tag{15}$$

$$Q_{t+1}^B(s_t, a_t) = Q_t^B(s_t, a_t) + \eta[r_t + \gamma Q_t^A(s_{t+1}, b^*) - Q_t^B(s_t, a_t)] \tag{16}$$

where, a^* is the action with the maximum q-value in state s_{t+1} according to the Q^A function and b^* is the action with the maximum q-value in state s_{t+1} according to the Q^B function. The double estimator technique is unbiased which results in no overestimation of action values, since action evaluation and action selection is decoupled into two functions that use separate $\max$ function estimates of action values. In fact, thorough analysis of Double Q-learning in [58] shows that it sometimes might underestimate action values.
The Double Q-learning algorithm was adapted for large state-action spaces in [59] by forming the Double DQN method in a similar way as DQN. The two Q-functions (Q^A and Q^B) can be parameterised by two sets of weights θ and θ'. At each step, one set of weights θ is used to update the greedy policy and the other θ' to calculate it's value. For Double DQN, equation 10 can be written as:

$$y = r_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a_t; \theta_i); \theta_i') \tag{17}$$

The first set of weights θ are used to determine the greedy policy just like in DQN. But, in Double DQN, the second set of weights θ' is used for an unbiased value estimation of the policy. Both sets of weights can be updated symmetrically by switching between θ and θ'.
The target value network in DQN can be used as the second Q-function instead of introducing an additional network. So, the weights at the i^{th} iteration are used to evaluate the greedy policy and the weights at the previous iteration to estimate it's value. The update rule remains the same as DQN, while changing the target as:

$$y = r_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a_t; \theta_i); \theta_{i-1}) \tag{18}$$

Note that in both DQN and DDQN, the target network uses the parameters of the previous iteration $i-1$. However, to generalise, the target network can use parameters from the any previous $(i-k)^{th}$ iteration. Then, the target network parameters are updated periodically with the copies of the parameters of the online network.

E. Dueling DQN

In quite a few RL applications, it is sometimes unnecessary to estimate the value of each action. In many states, the choice of action has no consequence on the outcome. A new architecture for model-free Reinforcement Learning, called the dueling architecture, is proposed in [60]. The dueling architecture explicitly separates state values and action advantage values into two streams which share a common feature extraction backbone neural network. The architecture is similar to that of the DQN and DDQN architectures; the difference being that instead of a single stream of fully connected layers, there are two streams providing estimates of the value and state-dependent advantage functions. The two streams are combined at the end producing a single Q-function.
One stream outputs a scalar state value, while the other outputs an advantage vector having dimensionality equal to number of actions. Both the streams are combined at the end to produce the Q-function estimate. The combining module at the end can simply aggregate the value and advantage estimates as:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \mathcal{A}(s_t, a_t; \theta, \alpha) \tag{19}$$

where, θ are the parameters of the lower layers of the neural network (before streams are split); α and β are the parameters of the advantage and value function streams. However, such an aggregation of streams would require $V(s_t; \theta, \beta)$ to be replicated as many times as the dimensionality of $\mathcal{A}(s_t, a_t; \theta, \alpha)$. Also, value and advantage estimates cannot be uniquely recovered given the estimated Q-function.
One way of addressing these issues, proposed in [60], is to force the advantage function estimator to have zero value at the selected action. This aggregation is implemented in the combining module as:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \left(\mathcal{A}(s_t, a_t; \theta, \alpha) - \max_{a_{t+1} \in A} \mathcal{A}(s_t, a_{t+1}; \theta, \alpha)\right) \tag{20}$$

Now, for a chosen action (action with max Q-function), $a^* = \arg\max_{a_{t+1} \in A} Q(s_t, a_{t+1}; \theta, \alpha, \beta)$, putting in equation 20, we get $Q(s_t, a^*; \theta, \alpha, \beta) = V(s_t; \theta, \beta)$. Hence, the two streams can be uniquely recovered.

In [60], another way of aggregation is proposed which eliminates the $\max$ operator.

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \left(\mathcal{A}(s_t, a_t; \theta, \alpha) - \frac{1}{|A|} \sum_{a_{t+1}} \mathcal{A}(s_t, a_{t+1}; \theta, \alpha)\right) \quad (21)$$

where, $|A|$ is the number of actions. Even though value and advantage estimates are now off-target by a constant, this way of aggregation improves stability by capping the changes in the advantage estimates by their mean and enhances overall performance.
In this paper, we use above mentioned off-policy, model-free algorithms on our novel fire evacuation environment and significantly improve performance for each of the above methods by transferring tabular Q-learning knowledge of the building structure into these methods.

III. Fire Emergency Evacuation System

In this paper, we propose the first fire evacuation environment to train reinforcement learning agents and a new transfer learning based tabular Q-learning+DQN method that outperforms state-of-the-art RL agents on the proposed environment. The fire evacuation environment consists of realistic dynamics that simulate real-world fire scenarios. For such a complex environment, an out-of-the-box RL agent doesn't suffice. We incorporate crucial information in the agent before training it, like the shortest path to the exit from each room. The rest of the section explains the entire system in detail.

A. *The Fire Evacuation Environment*

We propose the first benchmark environment for fire evacuation to train reinforcement learning agents. To the best of our knowledge, this is the first environment of it's kind. The environment has been specifically designed to simulate realistic fire dynamics and scenarios that frequently arise in real world fire emergencies. We have implemented the environment in the OpenAI gym format [11], to facilitate further research.
The environment has a graph based structure to represent a building model. Let $G = (V, E)$ be an undirected graph, such that $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices that represents n rooms and hallways and $E = \{e_1, e_2, ..., e_m\}$ is a set of edges that represents m paths connecting the rooms and hallways. A simple fire evacuation environment consisting of 5 rooms and paths connecting these rooms is shown in Fig. 2.
To represent the graph consisting of rooms, hallways and connecting paths, we use the adjacency matrix M_A. It is a square matrix consisting of elements $[0, 1]$ that indicate whether a pair of vertices is connected by an edge or not. The adjacency matrix is used to represent the structure of the graph and check the validity of actions performed by the agent. The adjacency matrix for the building model in Fig. 2 is given by:

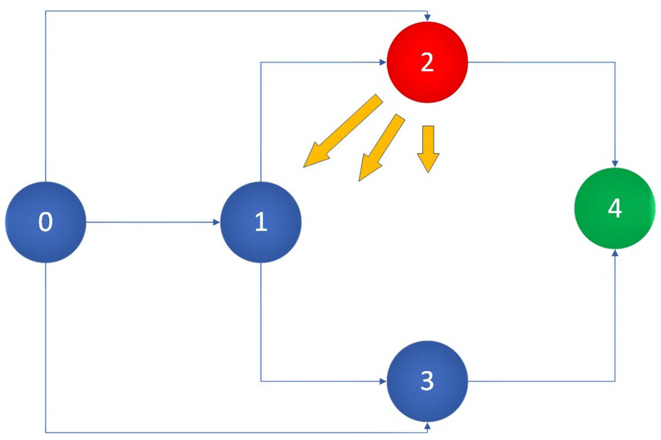


Fig. 2: A Simple Fire Evacuation Environment
The red vertex indicates fire in that room and the green vertex is exit. The orange arrows show the fire spread direction (more towards 1 compared to 3).

$$M_A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The environment dynamics are defined as follows:

*a) **State**:* Each vertex v_i of the graph represents a room and each room is associated with an integer N_i, which is the number of people in that room. The state of the environment is given by a vector consisting of the number of people in each room $S = [N_1, N_2, ..., N_n]$. To force the RL agent to learn the environment dynamics by itself, the environment doesn't provide any other feedback to the agent apart from the state (number of people left in each room) and the reward.

*b) **Action**:* An agent performs an action by moving a person from one room to the other and the state is updated after every valid action. Therefore, the action space is discrete. To keep things simple, we restrict the agent to move one person from one room at a time step. The agent can move a person from any room to any other room at any time step, even if the rooms aren't connected to each other by a path. So, the number of possible actions at each step is n^2.
This action space is necessary so that the agent can easily generalize to any graph structure. Also, this enables the agent to directly select which room to take people from and which room to send people to, instead of going through each room in a serial manner or assigning priorities.
When the agent selects an action, where there is no path between the rooms, the agent is heavily penalized. Due to this unrestricted action space and penalization, the agent is able to learn the graph structure (building model) with sufficient training and only performs valid actions at the end. The adjacency matrix is used to check the validity of actions.
Note that our graph based fire evacuation environment has n^2 possible actions (even though many of them are illegal moves and incur huge penalties), where n is the number of rooms.

Even for a small toy example of $n = 5$ rooms, the total number of possible actions is 25, which is a lot more than almost all of the OpenAI gym environments and Atari game environments [11].

*c) **Reward**:* We design a reward function specifically suited for our environment. We use an exponential decay function to reward/penalize the agent depending on the action it takes and to simulate fire spread as well. The reward function looks like this:

$$r(v_j, t) = -[d(v_j, t)]^t \tag{22}$$

where, t is the time step, v_j is the room where a person is moved to and $d(.)$ is the degree of fire spread for a room. $d(.)$ returns a positive number and if a room has a higher value of degree of fire spread, that means that fire is spreading more rapidly towards that room. We explicitly assign degrees to each room using a degree vector $D = [d(v_1, t), d(v_2, t), ..., d(v_n, t)]$, where the maximum value belongs to the room where the fire is located.
Using such a reward function ensures the following: Firstly, the reward values drop exponentially every time step as the fire increases and spreads. Secondly, the reward of an action depends on the room where a person is moved to. The reward function will penalize an action more heavily if a person is moved to a more dangerous room (higher degree of fire spread towards that room). This is because the function yields more rapidly decaying negative rewards. Lastly, the function yields a negative reward for every action which forces the agent to seek the least number of time-steps. The reward for reaching the exit is a constant $[r(v_j = exit, t) = +10]$.

*d) **Fire Location(s) and Exit(s)**:* The room where the fire occurs is given the highest degree, hence the maximum penalty for entering. The direction of fire spread is randomly decided and the degrees are assigned accordingly. The degrees are updated gradually to simulate fire spread.

$$d(v_j, t+1) = d(v_j, t) + \delta_j; \quad \forall v_j \quad in \quad V \tag{23}$$

where, δ_j is a small number ($0 \leq \delta \leq 1$) associated with v_j. δ is assigned to each room according to fire spread direction. So, δ can be used to determine fire spread direction, since higher value of δ for a room means that fire is spreading towards that room more rapidly.
As shown in Fig. 2, the fire spread is randomly and independently decided for all rooms v_j. The exit is also treated like a room. The only difference being that the agent gets a positive reward for moving people to the exit. The number of people at the exit is reset to zero after every action. The rooms which are exits are stored in a vector $\mathcal{E}$.

*e) **Bottleneck**:* Probably one of the most important feature in our proposed fire evacuation environment that enhances realism is the bottlenecks in rooms. We put an upper limit on the number of people that can be in a room at a time step. This restriction ensures congestion control during evacuation, which has been a huge problem in emergency situations. The bottleneck information is not explicitly provided to the agent, instead the agent learns about this restriction during training, since a negative reward is received by the agent if the number of people in a room exceed the bottleneck value. The bottleneck $\mathcal{B}$ is set to 10 in our experiments.

*f) **Uncertainty**:* To take into account uncertain behaviour of the crowd and introduce stochasticity in the environment, a person moves from one room to the other with probability $1 - p$. This means that an action a_t, selected by the agent at time-step t, is performed with probability $1-p$ or ignored with probability p. If the action is ignored, then there is no change in the state, but the reward received by the agent is as if the action was performed. This acts like a regularizing parameter and due to this, the agent is never able to converge to the actual global minimum. In our experiments, the uncertainty probability p is kept at 0.1.

*g) **Terminal Condition**:* The terminal/goal is reached once there are no people in any of the rooms $[\sum_{i=1}^{n} N_i = 0]$. The pseudocode for the proposed environment is given in Algorithm 1. From Algorithm 1, we can see that a heavier

```
Environment variables: $M_A$, $\mathcal{B}$, $\mathcal{E}$, $D$, $p$
Input: $S = [N_1, N_2, ...N_n]$
$t = 0$;
while not Terminal do
    $t = t + 1$;
    $a = agent.action(S)$;
    $v_i = a\%n$;
    $v_j = a/n$;
    if $p \geq random.uniform(0, 1)$ then
        $r = -D[v_j]^t$;
    end
    else
        if $SUM(S) == 0$ then
            Terminal;
        end
        else if $M_A[v_i, v_j] == 1$ and $v_j$ in $\mathcal{E}$ then
            $r = +10$;
            $S[v_i] = S[v_i] - 1$;
        end
        else if $M_A[v_i, v_j] == 0$ then
            $r = -2(\max(D)^t)$;
        end
        else if $S[v_j] \geq \mathcal{B}$ then
            $r = -0.5(\max(D)^t)$;
        end
        else
            $r = -D[v_j]^t$;
            $S[v_i] = S[v_i] - 1$;
            $S[v_j] = S[v_j] + 1$;
        end
    end
    Update $D$ according to $\delta$
end
```
Algorithm 1: Fire Evacuation Environment Pseudocode

penalty is received by the agent for an illegal move compared to bottleneck restriction violation and moving towards fire. In a way, rewards are used to assign priorities to scenarios. It can easily be changes if needed.

***Pretraining Environment**:* We create two instances of our environment: one for fire evacuation and the other for shortest path pretraining. For the pretraining instance, we consider only

the graph structure and the aim is to get to the exit from every room in the minimum number of time-steps.
The pretraining environment consists of the graph structure only, i.e. the adjacency matrix M_A. The pretraining environment doesn't contain fire, the number of people in each room or bottlenecks. The rewards are static integers: -1 for every path to force the agent to take minimum time-steps, -10 for illegal actions (where there is no path) and +1 for reaching the exit. The agent is thus trained to incorporate shortest path information of the building model.
The pseudocode for the pretraining environment is given in Algorithm 2. The procedure is repeated until the agent

```
Environment variables: $M_A$, $\mathcal{E}$
Input: $V = [v_1, v_2, ...v_n]$
$t = 0$;
$s = RandomSelection(V)$;
while not Exit do
    $t = t + 1$;
    $a = agent.action(s)$;
    $v_i = a\%n$;
    $v_j = a/n$;
    if $M_A[v_i, v_j] == 1$ and $v_j$ in $\mathcal{E}$ then
        $r = +1$;
        Exit;
    end
    else if $M_A[v_i, v_j] == 0$ then
        $r = -10$;
    end
    else
        $r = -1$;
        $s = v_j$;
    end
end
```

Algorithm 2: Shortest Path Pretraining Environment Pseudocode

converges to the shortest path from any room to the exit.

C

B. Similarities and Differences with Other Environments

The fire evacuation environment is implemented in the OpenAI gym format [11], to enable future research on the topic. OpenAI gym environments consists of four basic methods: *init*, *step*, *render* and *reset*. Our environment consists of the same four methods.
The *init* method consists of the initialization conditions of the environment. In our case, it contains the action space size, A, the state space size, $|S|$, the starting state S which is an array consisting of the number of people in each room (vertex), the adjacency matrix of the graph based building model, M_A and the fire location(s), F. The *reset* method simply sets the environment back to the initial conditions.
The *step* method is like the Algorithm 1, without the while loop. The *step* method takes in the action a_t performed at time-step t as the argument and returns the next state s_{t+1}, boolean variable for terminal T (indicating whether the terminal state was reached with the action performed or not) and the reward r_{t+1} for performing the action. The next state, reward and terminal depend on the conditions of the environment as shown in Algorithm 1. The *render* method simply returns the current state s_t.
The pretraining environment instance has the same structure of methods. The only difference is in the *step* method, shown in Algorithm 2 excluding the while loop, where the reward system is changed with fewer conditions and the state S is represented as the set of empty vertices (rooms with no people) of the graph.
Even though our environment might have the same structure as any OpenAI gym environment, it differs a lot in functionality from other environments or any game-based environments. In some ways, it might look like the mouse maze game in which the player (mouse) needs to reach the goal (cheeze) in the least possible steps through a maze. But, it is drastically different in many ways:

- Our environment is a graph based environment with much less connectivity then the maze environment, which makes finding the optimal route difficult.
- The optimal path(s) might change dynamically from one episode to the next or within a few time-steps due to fire spread and uncertainty in the fire evacuation environment, while the optimal path(s) for the mouse maze game remains the same.
- All the people in all the rooms must be evacuated to the nearest exit in the minimum number of time-steps under dynamic and uncertain conditions with bottlenecks, whereas in the mouse maze environment an optimal path only from the starting point to the goal needs to be found.
- The fire evacuation problem is a problem in which multiple optimal paths for all people in all rooms must be found while avoiding penalizing conditions like fire, bottlenecks and fire spread, whereas the mouse maze problem is a simple point-to-point problem.
- The mouse maze environment is static and lacks any variations, uncertainty or dynamics. On the other hand, the fire evacuation environment is dynamic, variable and uncertain.
- In the maze environment, the shortest path to the goal state is always the best. But, in the fire evacuation environment, even though the DQN agent is pretrained on the shortest path information, the shortest path to the exit might not be the best due to fire, fire spread and bottlenecks.
- The fire evacuation environment has a much larger action space n^2 than the maze environment (four actions: up, down, left, right) because all actions can be performed even if they are illegal (which will yield high penalties) to make the RL agent learn the building structure (graph model).
- Finally, a graph is a much better way to model a building's structure than a maze, since connectivity can be better described with a graph rather than a maze. It's what graphs were made for, to depict the relationships (connections) between entities (vertices).

Hence, the fire evacuation problem is a much more complex and dramatically different problem than the mouse maze problem or any other game based problem. Even the Go game has $19 \times 19 + 1$, i.e, 362 possible actions, whereas the fire evacuation environment has n^2 possible actions, i.e., as the number of rooms increase, the possible actions increase exponentially (although the Go game rules are quite complex to interpret by an RL agent).

C. *Q-matrix Pretrained Deep Q-Networks*

For the proposed graph based fire evacuation environment, we also present a new reinforcement learning technique based on the combination of Q-learning and DQN (and its variants). We apply tabular Q-learning to the simpler pretraining environment, with a small state space, to learn the shortest paths from each room to the nearest exit. The output of this stage is an $n \times n$ Q-matrix which contains q-values for state-action pairs according to the shortest path.

This Q-matrix is used to transfer the shortest path information to the DQN agent(s). This is done by pretraining the agent's neural network by deliberately overfitting it to the Q-matrix. After pretraining, the neural network weights have the shortest path information incorporated in them. Now, the agent is trained on the complete fire evacuation environment to learn to produce the optimal evacuation plan.

The main purpose of using such a strategy of training an agent by pretraining it first is to provide the agent with vital information about the environment beforehand, so that it doesn't have to learn all the complexities of the environment altogether. Since, after pretraining, the agent knows the shortest paths to the nearest exits in the building, dealing with other aspects of the environment like fire, fire spread, number of people and bottlenecks is made easier.

We provide two instances of our environment: simpler shortest path pretraining instance and complex fire evacuation instance. First, the agent is pretrained on the simpler instance of the environment (for shortest path pretraining) and then trained on the more complex instance (for optimal evacuation). This approach of training the agent on a simpler version of the problem before training it on the actual complex problem is somewhat similar to curriculum learning [61].

We also add a small amount of noise or offset to the Q-matrix produced by training on the pretraining environment instance. This is done by adding or subtracting (depending on the q-value) a small σ to each element of the Q-matrix.

$$Q(s,a) = \begin{cases} Q(s,a) + \sigma, & \text{if } Q(s,a) \leq 0 \\ Q(s,a) - \sigma, & \text{if } Q(s,a) > 0 \end{cases}$$

where, σ can be thought of as a regularization parameter, which is set to 10 in our experiments. Adding noise to the Q-matrix is necessary because we don't want the DQN agent to just memorize all the paths and get stuck at a local minimum. The actual fire evacuation instance is complex, dynamic and has uncertainty which means that an optimal path at time-step t might not be the optimal path at time-step $t + k$. The hyperparameter σ acts as a regularizer.

Note that we add σ if the element of the Q-matrix is negative or zero and subtract σ if the element is positive. This is done to offset the imbalance between good and bad actions. If we just add or subtract σ then the relative difference between q-values would remain the same. Conditional addition or subtraction truly avoids the DQN agent from being biased to a particular set of actions leading to an exit.

Even though pretraining adds some overhead to the system, there are several advantages including:

- **Better Conditioning** Pretraining provides the neural network with a better starting position of weights for training compared to random initializations.
- **Faster Convergence** Since the neural network weights are better conditioned due to pretraining, training starts closer to the optimum and hence rate of convergence is faster.
- **Crucial Information** Especially in the case of fire evacuation, pretraining with shortest path information provides the agent with crucial information about the environment before training begins.
- **Increased Stability** As pretraining restricts the weights in a better basin of attraction in the parameter space, the probability of divergence is reduced which makes the model stable.
- **Fewer number of updates** As the weights are near the optimal on the error surface, the number of updates required to reach the optimum is lower, which results in fewer memory updates and requiring less training epochs.

The pseudocode for the proposed Q-matrix pretrained DQN algorithm is given in Algorithm 3. The algorithm 3 consists of 3 functions: $Q_{learning}()$ for tabular Q-learning on the pretraining environment instance for finding optimal q-values for shortest path from each room to the nearest exit; $Agent_{pretrain}()$ for overfitting the shortest path Q-matrix to incorporate the information in the DQN Agent's network; $Main()$ for using the pretrained DQN Agent to learn the optimal evacuation plan by training it on the fire evacuation environment.

Modifying the final training part to include Double DQN and Dueling DQN Agents is straightforward.

D. *Pretraining Convergence*

The paper [25] thoroughly analyses and proves conditions where task transfer Q-learning will work. We use the proved propositions and theorems from [25] to show that pretraining works in our case.

Let the pretraining instance and the fire evacuation instance be represented by two MDPs: $M_1 = \langle S, A, R_1, P_1, \gamma_1 \rangle$ for pretraining instance and $M_2 = \langle S, A, R_2, P_2, \gamma_2 \rangle$ for fire evacuation instance. So, according to proposition 1 in [25]:

$$\Delta(M_1, M_2) = \parallel Q_1^* - Q_2^* \parallel \triangleq \frac{\parallel R_1 - R_2 \parallel_\infty}{1 - \gamma'} + \frac{\gamma'' \parallel R' \parallel_\infty}{(1 - \gamma'')^2} \parallel P_1 - P_2 \parallel_\infty + \frac{|\gamma_1 - \gamma_2|}{(1 - \gamma_1)(1 - \gamma_2)} \parallel R'' \parallel_\infty \tag{24}$$

where $\Delta(M_1, M_2)$ is the distance between MDPs and Q_1^* and Q_2^* are the corresponding optimal Q-functions. In our case, $\gamma_1 = \gamma_2$ and $P_1 = P_2 = 1$ since our environments are deterministic MDPs, i.e., taking an action a at state s will always lead to a specific next state s' and no other state, with

```
Environment instances: $Pretraining\_Env()$,
  $Fire\_Evacuation()$
Environment variables: $M_A$, $\mathcal{B}$, $\mathcal{E}$, $D$, $V$, $S$
$S_P = Shortest\_Path()$;
$F_E = Fire\_Evacuation()$;
Function $Q_{learning}()$
    for $i \leftarrow 0$ to $T_Q$ do
        $s_i = S_P.state()$;
        while not terminal do
            if $Random(0,1) < \epsilon$ then
                $a_i = Random\_Action()$;
            end
            else
                $a_i = \arg\max(Q_i[s_i])$;
            end
            $s_{i+1}, r_i, terminal = agent.act(a_i)$;
            Update $Q_i(s_i, a_i)$ using eq 8;
            $s_i = s_{i+1}$;
        end
    end
    return $Q^*$
End Function
Function $Agent_{pretrain}()$
    $s_0 = F_E.empty\_state()$;
    for $j \leftarrow 0$ to $T_{Pretrain}$ do
        $O_j = agent.predict(s_0)$;
        $L(\theta_j) = \frac{1}{2}(O_j - Q^*)^2$;
        $\theta_{j+1} = \theta_j + \eta \nabla L(\theta_j)$;
    end
    return $\theta^*$
End Function
Function $Main()$
    $DQN = load\_weights(\theta^*)$;
    for $t \leftarrow 0$ to $T$ do
        $s_t = F_E.state()$;
        while not terminal do
            $s_{t+1}, r_t, terminal = DQNAgent.act(s_t)$;
            Train the DQNAgent by:
              Calculate $L_t(\theta_t)$ using eq. 11;
              Calculate $\nabla_{\theta_t} L_t(\theta_t)$ using eq. 12;
              Update weights: $\theta_{t+1} = \theta_t + \eta \nabla L_t(\theta_t)$;
            $s_t = s_{t+1}$;
        end
    end
    return $DQNAgent$
End Function
```

Algorithm 3: Q-Matrix Pretrained DQN

probability $p(s'|s,a) = 1$. This makes the second and third term of equation 24 to zero. So, the distance between the two instances of our environment is reduced to the first term only. So, according to proposition 1, if the distance between two MDPs is small, then the learned Q-function from the pretraining task is closer to the optimal of the fire evacuation task compared to random initializations and hence helps in convergence to an optimum and improves the speed of convergence. Also, convergence is guaranteed according to theorem 4 in [25], if the safe condition is met:

$$\frac{(1-\gamma)\Delta(M_1, M_2)}{BE(Q(s,a))} \leq 1 \tag{25}$$

where, $BE(Q(s,a)) = |[R + \gamma Q(s', a')] - Q(s,a)|$ is the Bellman error. In our case, $\Delta(M_1, M_2)$ is small and that multiplied by $1-\gamma$, which is less than 0.1 since $\gamma > 0.9$, is less than 1. For our case, this seems obvious since the two MDPs are instances of the same MDP. This means that convergence is guaranteed, as long as the shortest path Q-matrix obtained from the pretraining environment converges.

Now, to prove that our method has guaranteed convergence, we need to prove that the Q-matrix is able to capture the shortest path information accurately.

E. Convergence Analysis of Q-learning for finding shortest path

The guarantee of convergence for Q-learning has been discussed and proved in many different ways and for general as well as unique settings [28], [62]. The convergence of Q-learning is guaranteed, while using the update rule given in equation 8, if the learning rate η is bounded between $0 \leq \eta < 1$ and the following conditions hold:

$$\sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} [\eta_t]^2 < \infty \tag{26}$$

Then, $Q_t(s,a) \longrightarrow Q^*(s,a)$ as $t \longrightarrow \infty$, $\forall s, a$, with probability 1. This means that for the learning rate conditions to hold with the constraint $0 \leq \eta < 1$, all state-action pairs must be visited an infinite number of times. Here, the only complication is that some state-action pairs might never be visited.

In our pretraining environment, which is an episodic task, we can make sure that all state-action pairs are visited by starting episodes at random start states which is shown in Algorithm 2. Apart from this we use an ϵ-greedy exploration policy to explore all state-action pairs. The initial value of ϵ and the decay rate are set according to the size of the graph based environment.

We run Q-learning on the pretraining environment for ~ 1000 episodes so ensure that the Q-matrix converges to $Q^*(s,a)$. Since we have an action space of 25 actions for 5 rooms, running for more episodes is convenient. But, for large building models (8281 actions for the large real world building scenario, in Section 5), running for many episodes could become computationally too expensive. So, we use a type of early stopping criteria, in which we stop training the Q-matrix if there is a very small change in it's elements from one episode to the next.

However, as we shall see in Section 5, that we do not require early stopping at all. We are able to reduce the action space drastically and hence the Q-matrix can be trained in the same way as it was trained for smaller action spaces.

In [63], the proof of convergence of Q-learning is given for stochastic processes, but in our case, the environment is deterministic. Also, in [64], a more general convergence proof for Q-learning is provided using convergence properties of stochastic approximation algorithms and their asynchronous versions. The asymptotic bounds for the error $\triangle_t(s,a) = |Q_t(s,a) - Q^*(s,a)|$ has been shown to be bound by the number of visits to state-action pairs and t:

$$\triangle_t(s,a) \propto \frac{1}{t^{R(1-\gamma)}} \tag{27}$$

where, $R = \frac{min_{s,a}P(s,a)}{max_{s,a}P(s,a)}$ and $P(s,a)$ is the sampling probability of (s,a). So, it is necessary to run the Q-learning algorithm for as many episodes as possible. Hence, we device a strategy to reduce the action space for large discrete action spaces, which are as a result of real world building models, so that it becomes feasible to train the Q-matrix for a large number of episodes.

F. Discussion on alternative Transfer Learning techniques

There are a few ways of pretraining an agent, some of which have been discussed and evaluated in [65]. A naive approach would be to preload the experience replay memory with demonstration data before hand. This method, however, isn't actually pretraining. The agent trains normally with the benefit of being able to learn good transitions immediately.

Our method of pretraining beckons the question of pretraining the agent's network directly. Pretraining a DQN network's weights on the pretraining environment would require more time compared to tabular Q-learning. The DQN would require more time to converge. Also, in the next step where the Q-matrix is used as a fixed output to train the network's weights to overfit on the q-values requires much less time. Also, for a smaller state space (like the pretraining environment) tabular Q-learning is much more efficient than DQN. The total time taken for pretraining using the proposed method is $5.15s$ ($3.1s$ for tabular Q-learning and $2.05s$ for overfitting the agent's weights on the Q-matrix) compared to $9.8s$ for pretraining DQN directly. It's because using the direct pretraining method would effectively require the DQN to be trained twice overall (once on the pretraining environment and then on the fire evacuation environment), which is inefficient and computationally expensive.

Also, this complexity will grow exponentially when we train it on a large real world building model, which is shown in Section 5. For an environment with $n = 91$ rooms and 8281 actions, training a DQN agent twice would be extremely inefficient and computationally infeasible, due to the size of the neural network and computations required and the expense of backpropagation. Whereas, training the Q-matrix would only require computing equation 8.

One of the most successful algorithms in pretraining deep reinforcement learning is the Deep Q-learning from Demonstrations (DQfD) [66], [67]. It pretrains the agent using a combination of Temporal Difference (TD) and supervised losses on demonstration data in the replay memory. During training, the agent trains its network using prioritized replay mechanism between demonstration data and interactions with the environment to optimize a complex combination of four loss functions (Q-loss, n-step return, large margin classification loss and L2 regularization loss). The DQfD uses a complex loss function and the drawback of using demonstration data is that it isn't able to capture the complete dynamics of the environment as it covers a very small part of the state space. Also, prioritized replay adds more overhead. Our approach is far simpler and because we create a separate pretraining instance to incorporate essential information about the environment instead of the full environment dynamics, it is more efficient than demonstration data.

IV. Experiments and Results

We perform unbiased experiments on the fire evacuation environment and compare our proposed approach with state-of-the-art reinforcement learning algorithms. We test different configurations of hyperparameters and show the results with best performing hyperparameters for these algorithms on our environment. The main intuition behind using Q-learning pretrained DQN model was to provide it with important information before hand, to increase stability and convergence. The results confirms our intuition empirically.

The Agent's Network: Unlike the convolutional neural networks [31] used in DQN [51], [52], DDQN [58], [59] and Dueling DQN [60], we implement a fully connected feedforward neural network. The network configuration is given in Table 1. The network consists of 5 layers. The ReLU function [68] is used for all layers, except the output layer, where a linear activation is used to produce the output.

C

Environment: The environment given in Fig. 2 is used for all unbiased comparisons. The state of the environment is given as : $S = [10, 10, 10, 10, 0]$ with bottleneck $\mathcal{B} = 10$. All rooms contain 10 people (the exit is empty), which is the maximum possible number of people. We do this to test the agents under maximum stress. The fire starts in room 2 and the fire spread is more towards room 1 than room 3 (as shown in Fig. 2 with orange arrows). Room 4 is the exit. The total number of actions possible for this environment is 25. So, the agent has to pick one out of 25 actions at each step.

Training: The Adam optimizer [56] with default parameters and a learning rate η of 0.001 is used for training for all the agents. Each agent is trained for 500 episodes. Training was performed on a 4GB NVIDIA GTX 1050Ti GPU. The models were developed in Python with the help of Tensorflow [69] and Keras [70].

Implementation: Initially, the graph connections were represented as 2D arrays of the adjacency matrix M_A. But, when the building model's graphs get bigger, the adjacency matrices become more and more sparse, which makes the 2D array representation inefficient. So, the most efficient and easiest way to implement a graph is as a dictionary, where the keys represent rooms and their values are an array that lists all the rooms that are connected to it.

$$dict_{graph} = \{room_i : [room_j; \quad \forall j \quad in \quad M_{A_{i,j}} = 1]\}$$

Comparison Graphs: The comparison graphs shown from Fig. 3 to Fig. 10 have the total number of time-steps required for complete evacuation for an episode on the y-axis and the

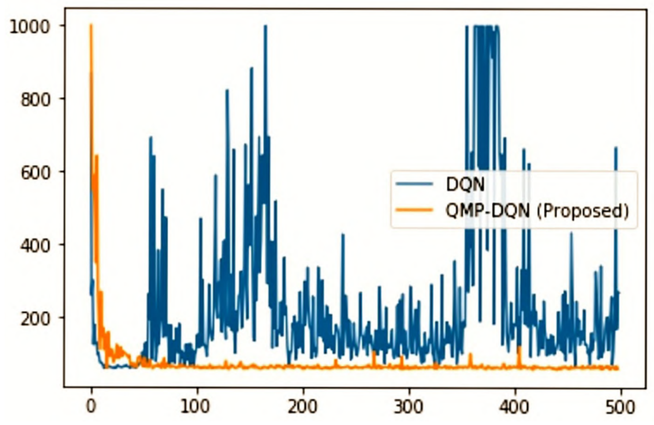


Fig. 3: Q-matrix pretrained DQN vs DQN

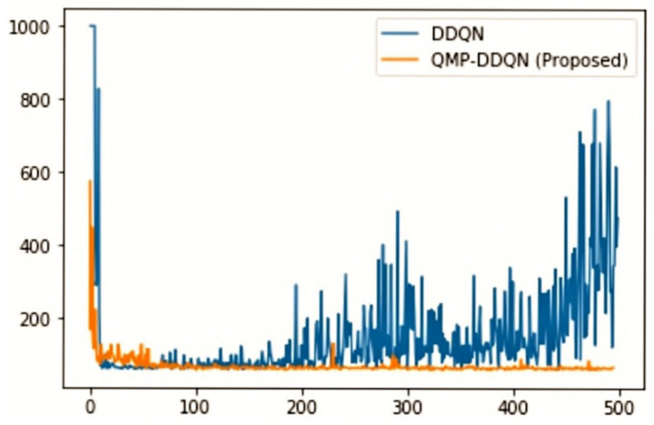


Fig. 4: Q-matrix pretrained DDQN vs DDQN

number of episodes on the x-axis. The comparisons shown in the graphs are different runs of our proposed agents with exactly the same environment settings used for all the other agents as well.

We first compare the Q-matrix pretrained versions of the DQN

TABLE I: Network Configuration

Type	**Size**	**Activation**
Dense	128	ReLU
Dense	256	ReLU
Dense	256	ReLU
Dense	256	ReLU
Dense	No. of actions	Linear

and its variants with the original models. The graph based comparisons between models consists of number of time-steps for evacuating all people on the y-axis and episode number on the x-axis. We put an upper-limit of 1000 time-steps for an episode due to computational reasons. The training loop breaks and a new episode begins once this limit is reached.

The graph comparing DQN with our proposed Q-matrix pretrained DQN (QMP-DQN) in Fig. 3 shows the difference in their performance on the fire evacuation environment. Although the DQN reaches the optimal number of time-steps quickly, it isn't able to stay there. The DQN drastically diverges from the solution and is highly unstable.

It's the same case with DDQN (Fig. 4) and Dueling DQN (Fig. 5), which, although perform better that DQN with less fluctuations and spend more time near the optimal solution. Our results clearly shows a big performance lag compared to the pretrained versions. As these results suggest that pretraining ensures convergence and stability. We show that having some important information about the environment prior to training reduces the complexity of the learning task for an agent.

The original Q-learning based models aren't able to cope with the dynamic and stochastic behaviour of the environment. And since they don't posses pretrained information, their learning process is made even more difficult. Table 2 displays a few

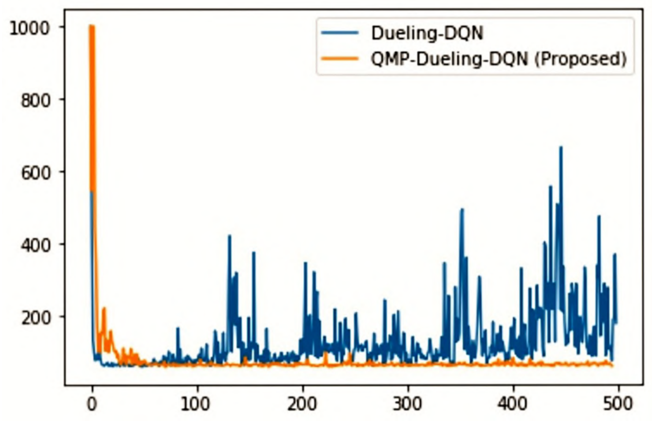


Fig. 5: Q-matrix pretrained Dueling DQN vs Dueling DQN

numerical results, comparing DQN, DDQN and Dueling DQN, with and without the Q-matrix pretraining on the basis of average number of time-steps for all 500 episodes, minimum number of time-steps reached during training and the training time per episode.

As it was also clear from the Figs. 3, 4 and 5, the average number of time-steps is greatly reduced with pretraining, as it makes the models more stable by reducing variance. Based on the environment given in Fig. 2, the minimum possible number of time-steps is 60. All the DQN based models are able to come close to this, but pretraining pushes these models further and achieves the minimum possible number of time-steps. Even though the difference seems small, in emergency situations even the smallest differences could mean a lot at the end. The training time is also reduced with pretraining, as the number of time-steps taken during training is reduced and pretrained models get a better starting position nearer to the optimum.

Next, we make comparisons between our proposed approach and state-of-the-art reinforcement learning algorithms. For these comparisons, we use the Q-matrix pretrained Dueling DQN model, abbreviated QMP-DQN. We also compare it with

TABLE II: Performance

Model	Average Time-Steps	Average Time-Steps with Pretraining	Minimum Time-Steps	Minimum Time-Steps with Pretraining	Training Time (per episode)	Training time with Pretraining (per episode)
DQN	228.2	**76.356**	63	**61**	10.117	**6.87**
DDQN	134.62	**71.118**	61	**60**	12.437	**8.11**
Dueling DQN	127.572	**68.754**	61	**60**	12.956	**9.02**

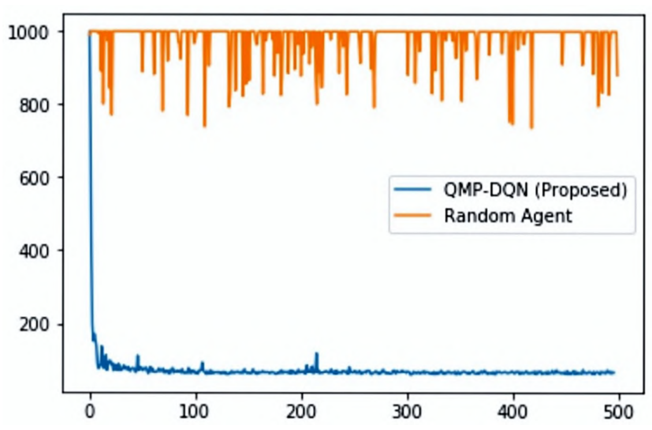


Fig. 6: Proposed method vs Random Agent

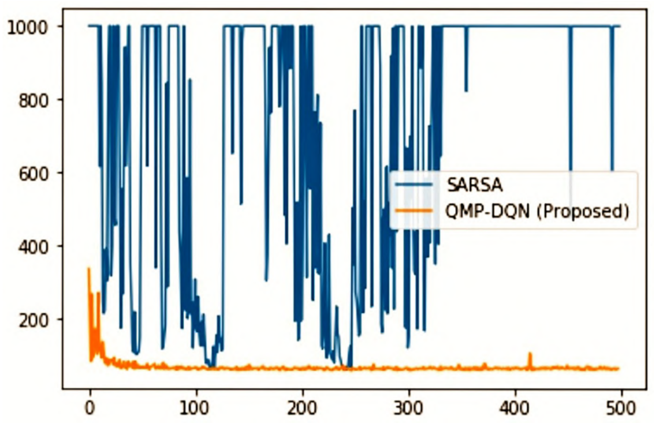


Fig. 7: Proposed method vs State-Action-Reward-State-Action method

a random agent, shown in Fig. 6. The random agent performs random actions at each step, without any exploration. The random agent's poor performance of 956.33 average time-steps shows that finding the optimal or even evacuating all the people isn't a simple task.

The State-Action-Reward-State-Action (SARSA) algorithm is an on-policy reinforcement learning agent introduced in [3]. While Q-learning follows a greedy policy, SARSA takes the policy into account and incorporates it into its updates. It updates values by considering the policy's previous actions. On-policy methods like SARSA have a downside of getting trapped in local minima if a sub-optimal policy is judged as

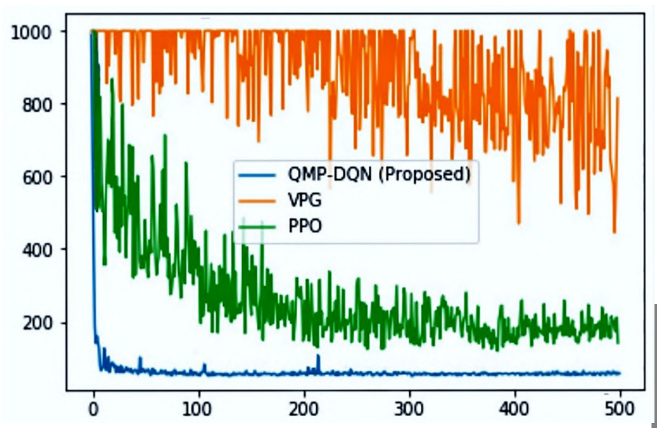


Fig. 8: Proposed method vs Policy based methods (PPO and VPG)

the best. On the other hand, off-policy methods like Q-learning are flexible and simple as they follow a greedy approach. As it is clear from Fig. 7, that SARSA behaves in a highly unstable manner and isn't able to reach the optimal solution and shows high variance.

Policy gradient methods are highly preferred in many applications, however they aren't able to perform optimally on our fire evacuation environment. Since the optimal policy could change in a few time-steps in our dynamic environment, greedy action selection is probably the best approach. An evacuation path that seems best at a particular time step could be extremely dangerous after the next few time-steps and a strict policy of routing cannot be followed continuously due to fire spread and/or bottleneck. These facts are evident from Fig. 8, where we compare our approach to policy gradient methods like Proximal Policy Optimization (PPO) [71] and Vanilla Policy Gradient (VPG) [72]. Even though PPO shows promising movement, it isn't able to reach the optimum.

Another major type of reinforcement learning algorithms are the actor-critic methods. It is a hybrid approach consisting of two neural networks: an actor which controls the policy (policy based) and a critic which estimates action values (value based). To further stabilize the model, an advantage function is introduced which gives the improvement of an action compared to an average action used in a particular state. Apart from the previously mentioned shortcomings of using policy based methods on the fire evacuation environment, the advantage function would have high variance since the best action at a particular state could change rapidly leading to

TABLE III: Comparison with State-of-the-art RL Algorithms

Model	**Average Time-Steps**	**Minimum Time-Steps**	**Training Time (per episode)**
SARSA	642.21	65	19.709
PPO	343.75	112	16.821
VPG	723.47	434	21.359
A2C	585.92	64	25.174
ACKTR	476.56	79	29.359
Random Agent	956.33	741	-
QMP-DQN (Dueling DQN Backbone)	**68.754**	**60**	**9.02**

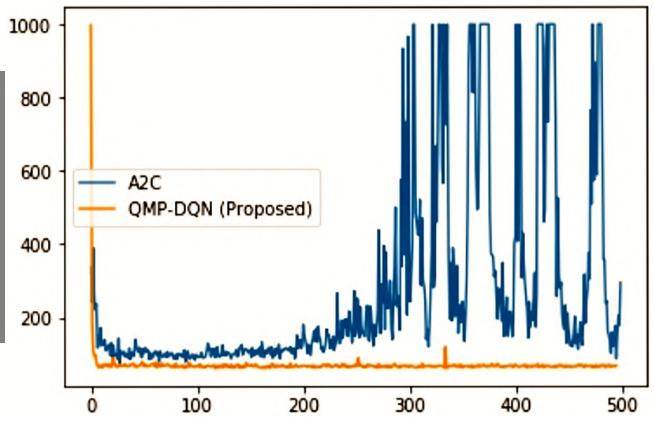


Fig. 9: Proposed method vs Synchronous Advantage Actor Critic method (A2C)

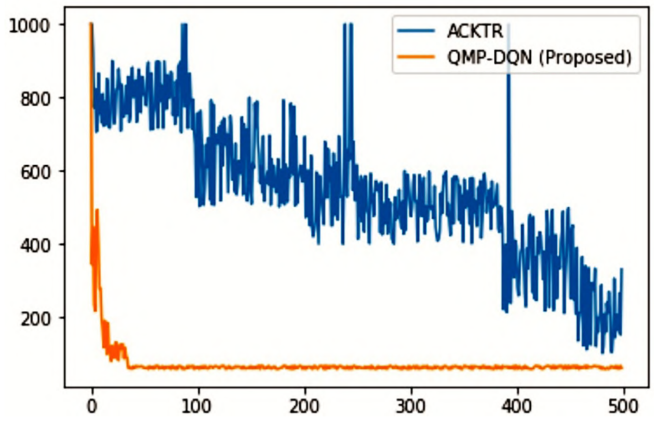


Fig. 10: Proposed method vs Actor Critic using Kronecker-Factored Trust Region (ACKTR)

unstable performance. This is apparent from Fig. 9, where we compare the synchronous advantage actor critic method (A2C) [73] with our proposed method. The A2C gives near optimal performance in the beginning but diverges and rapidly fluctuates.

We do not compare our proposed method with the asynchronous advantage actor critic method (A3C) [74], because A3C is just an asynchronous version of A2C, which is more complex as it creates many parallel versions of the environment and gives relatively the same performance, but is not as sample efficient as claimed in [75]. The only advantage of A3C is that it exploits parallel and distributed CPU and GPU architectures which boosts learning speed as it can update asynchronously. However, the main focus of this paper is not learning speed. Hence, we think that the comparison with A2C is sufficient for actor-critic models.

Probably the best performing Actor Critic based model is the ACKTR (Actor Critic with Kronecker-factored Trust Region) [76]. The algorithm based on applying trust region optimization using Kronecker-factored approximation, which is the first scalable trust region natural gradient method for actor critic models that can be applied to both continuous and discrete action spaces. The Kronecker-factored Approximate Curvature (K-FAC) [77], is used to approximate the Fisher Matrix to perform approximate natural gradient updates. We compare our method to the ACKTR algorithm, shown in Fig. 10. The results suggest that the ACKTR is not able to converge (within 500 episodes, due to slow convergence rate) and is susceptible to the dynamic changes in the environment as evident from the fluctuations. ACKTR is far too complex compared to our proposed method, which converges much faster and deals with the dynamic behaviour of the fire evacuation environment efficiently.

We summarize our results in Table 3. All the RL agents use the same network configuration mentioned in Table 1 for unbiased comparison. The training time for the QMP-DQN is much lower compared to other algorithms because pretraining provides it with a better starting point, so it requires less number of time-steps and memory updates to reach the terminal state. Also, SARSA and A2C come really close to the minimum number of time-steps, but as the average number of time-steps suggests, they aren't able to converge and exhibit highly unstable performance. Our proposed method,

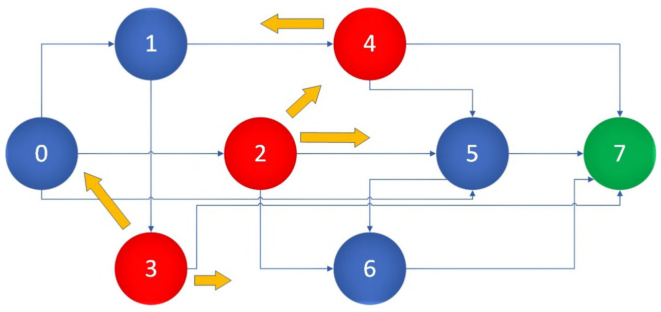


Fig. 11: A Multiple Fire Evacuation Environment

Q-matrix pretrained Dueling Deep Q-network gives the best performance on the fire evacuation environment by a huge margin.
Note that, in all the comparison graphs, our proposed method comes close to the global optimum, but isn't able to completely converge to it. This is because of the uncertainty probability p, which decides whether an action is performed or not and is set to 0.15. This uncertainty probability is used to map the uncertain crowd behaviour. Even though, p, does not allow complete convergence, it also prevents the model from memorizing an optimal path which might change as the fire spreads.

Multiple Fires Scenario

Now that we have shown that the proposed method is able to outperform state-of-the-art reinforcement learning algorithms, we test our model on a more complex and difficult environment setup. The environment configuration consists of multiple fires in different rooms and a more complex graph structure consisting of 8 rooms. The environment is shown in Fig. 10. The green node is the exit, the red nodes are the rooms where the fire is located and the orange arrows depict the direction of fire spread.
As we can see from Fig. 10, the fire spreads in different directions from different fire locations. This makes things especially difficult because as the fire spreads, the paths to the exit could be blocked. We do not change the configuration of our approach, except the output layer of the network, since the number of possible actions is 64 now. The State of the environment and Bottleneck given as input is: $S = [10, 10, 10, 10, 10, 10, 10, 0]$ and $\mathcal{B} = 10$.
We employ the Q-matrix pretrained Dueling DQN model. Fig. 11 shows the graphical results on the multiple fires scenario. The initial fluctuations are due to ϵ-greedy exploration. Since this configuration of the environment is bigger and more complex, the agent explores the environment a little longer.
As the results suggest from Fig. 11, the proposed model is able to converge very quickly. A few metrics for the proposed method on the multiple fires environment is given below:

- **Average number of time-steps:** 119.244
- **Minimum number of time-steps:** 110
- **Training time (per episode):** 15.628

Note that, there is a difference of ≈ 9 time-steps between the minimum number of time-steps and average number of time-steps. This is because the average of all 500 episodes is taken which includes the initial fluctuations due to exploration and the uncertainty probability p.

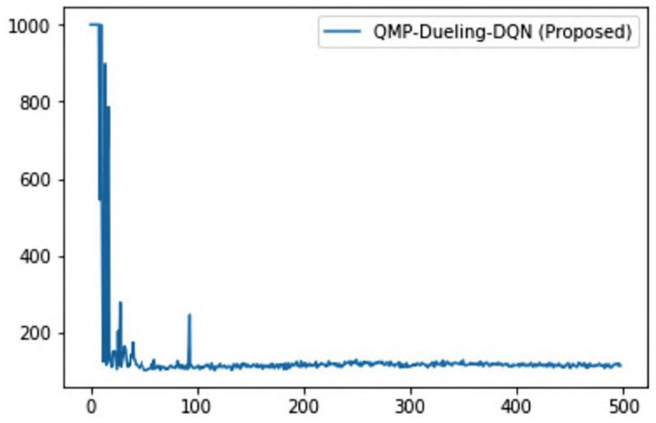


Fig. 12: Q-matrix pretrained Dueling DQN in Multiple Fire Scenario

V. Scalability: Large and Complex Real World Scenario - University of Agder Building

To prove that our method is capable of performing on large and complex building models, we simulate a real world building, i.e., the University of Agder A, B, C and D blocks, and perform evacuation in case of fire in any random room(s). This task is especially difficult because of the resulting complex graph structure of the building and the large discrete action space. We consider the A, B, C and D blocks which are in the same building. The total number of rooms in this case is $n = 91$, which means that the number of all possible actions is 8281. This discrete action space is many times larger than any other OpenAI gym environment or Atari game environments [11]. Even the Go game has $19 \times 19 + 1$, i.e., 362 actions.
Dealing with such a large action space would require a huge agent model or moving towards to a multi-agent approach and dividing the environment into subsets, with each sub-environment for each agent to deal with. These techniques for dealing with the large discrete action space would be computationally complex and difficult to implement for the fire evacuation environment.
Another way could be to use a policy gradient method which are much more effective in dealing with large action spaces compared to value based methods. But, dealing with such large action spaces would require an ensemble of neural networks and tree search algorithms like in [78] or extensive training from human interactions like in [79]. However, in a fire emergency environment we obviously can't have human interactions and we would like to solve the issue of large action space without having to use dramatically huge models. Also we saw in the previous section that even though PPO performs much better compared to other algorithms, it wasn't able to outperform our QMP-DQN methods.
In [80], a new method to deal with extremely large discrete action spaces ($\sim$1 million actions) was proposed. The novel

method, called the Wolpertinger policy algorithm, uses a type of actor-critic architecture, in which the actor proposes a proto-action in an action embedding space from which k most similar actions are selected using the k-nearest neighbour algorithm. These k actions are received by the critic which makes a greedy selection based on the learned q-values. This technique shows promising results, however, it is highly complex.

We propose a much simpler approach to deal with large number of actions. Our method consists of two stages: One-Step Simulation (OSS) of all actions resulting in an action importance vector A_I and then element-wise addition with the DQN output for training. We explain our method in the following subsections.

A. *One-Step Simulation and Action Importance Vector*

We make use of the pretraining environment instance shown in Algorithm 2 to calculate the action importance vector A_I, as shown in Algorithm 4. The $one\_step\_sim(s, k)$ function is implemented in the environment itself to enable the environment object to use the method and the function to use the environment variables.

The $one\_step\_sim(s, k)$ function simulates all possible actions for each state/room for one time-step in the pretraining environment. It stores all rewards received for these actions taken from room s and returns the k best actions for each room s which yield the k highest rewards.

The $one\_step\_sim(s, k)$ function is run for each room s in N, which is the total number of rooms in the environment. The equation $x[j] \longleftarrow env.one\_step\_sim(j, k) * N + j$, is used to convert the k best actions returned by $one\_step\_sim(s, k)$ function for all rooms s, into a single vector of actions. This is necessary because the DQN agent can take any appropriate action from any room at a particular time-step. So, it outputs a single vector consisting of Q-values for all actions at each time-step.

After we have a unique index for all selected actions in the environment, we form the action importance vector A_I by placing 0 at index l, if the l^{th} action is present in the vector x, which consists of all the k best actions for each room s, otherwise, a large negative number (like -9999) at index l.

The action importance vector can be though of as a fixed weight vector which contains weight 0 for good actions and a large negative weight for others. A_I is then added element-wise with the output of the DQN $\hat{Q}$ to produce the final output Q^* on which the DQN is trained on.

$$Q^* = \hat{Q} \oplus A_I \tag{28}$$

This makes the Q-values of the good actions to remain the same and reduces the Q-values of other actions to huge negative numbers. This method effectively reduces the action space from $O(N^2)$ to $O(kN)$, where $k \ll N$. In our experiments, we set the hyperparameter k as the maximum degree of vertices in the building model's graph, i.e. $k = 9$. So, in our model, the action space is effectively reduced from 8281 actions to 819 actions, which is a 90.1% decrease.

Hence, our complete method consists of shortest path pretraining using Q-matrix transfer learning and action space reduction by one-step simulation and action importance and finally DQN based model training and execution. The shortest path pretraining provides the model with global graph connectivity information and the one-step simulation and action importance delivers local action selection information.

```
**Environment instances:** $Pretraining\_Env()$
**Environment variables:** $N \longleftarrow$ Number of rooms
$env = Pretraining\_Env();$
**Function** $one\_step\_sim(s, k)$
    **for** $i$ **in** $action\_space$ **do**
        $s_{t+1}, r, terminal = env.step(i);$
        $rewards[i] \longleftarrow r;$
    **end**
    **return** $rewards.argsort(k);$
**End Function**
**for** $j$ **in** $N$ **do**
    $x[j] \longleftarrow env.one\_step\_sim(j, k) * N + j;$
**end**
**for** $l$ **in** $action\_space$ **do**
    **if** $l$ **in** $x$ **then**
        $A_I[l] = 0;$
    **end**
    **else**
        $A_I[l] = -9999;$
    **end**
**end**
```

Algorithm 4: One-Step Simulation and A_I

The action importance vector can also be thought of as an attention mechanism [37], [81]–[83]. Most of the attention mechanisms employ a neural network or any other technique to output an attention vector which is then combined with the input or an intermediate output to convey attention information to a model. Unlike these methods, our proposed model combines the action importance vector with the output of the DQN. This means that the current action selection is based on a combination of the Q-values produced by the DQN and the action importance vector, but the training of the DQN is impacted by the attention vector in the next iteration of training, as the final output of the i^{th} iteration is used as the label for training the model at the $i + 1^{th}$ iteration.

One major advantage of such an attention mechanism used in our method is that, since the graph based environment has a fixed structure, the attention vector needs to be calculated just once at the beginning. We test our method on the University of Agder (UiA), Campus Grimstad building with blocks A, B, C and D consisting of 91 rooms.

Note that, unlike the usual attention based models, we do not perform element-wise multiplication of the attention vector with the output of a layer. Instead, we add the attention vector because initially the DQN model will explore the environment and will have negative Q-values for almost all actions (if not all). This means that if we use a vector of ones and zeros for good and bad actions respectively and multiply element-wise with the output of a layer then, the Q-values of good actions will be copied as it is and the Q-value of other actions will

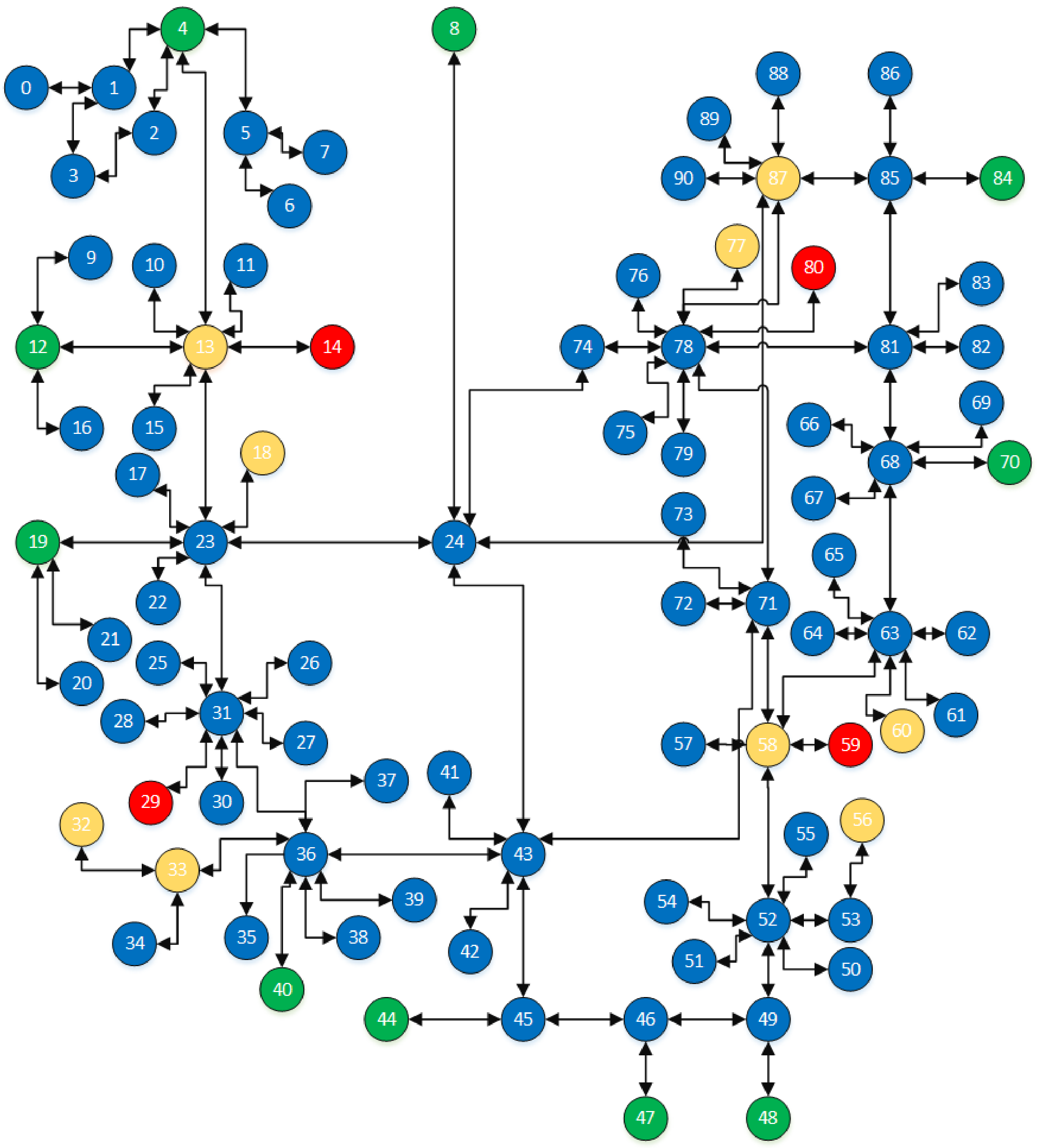


Fig. 13: University of Agder Graph
The red vertices indicate fire in that room and the green vertices are exits. The yellow vertices show the fire spread towards that room.

become zero. If the Q-value of good actions is negative in the beginning due to exploration (and lack of learning since it is the beginning of training), then the max function in the Q-value selection equation will select bad actions since they are zeros and good actions are negative. This will lead to catastrophic behaviour of the system and it will never converge. So, instead we use addition with zeros for good actions so that they remain the same and with large negative numbers for other actions so that their Q-values become so low that they are never selected.

B. *Fire Evacuation in the UiA building*

The graph for UiA's building model is based on the actual structure of the 2nd floor of blocks A, B, C and D[1]. The graph for the building model is shown in Fig. 13. It consists of 91 rooms (from room 0 to room 90) out of which there are 10 exits. We simulate the fire evacuation environment in which there are multiple distributed fires in rooms 14, 29, 59 and 80. The fire spread for each fire is individually simulated in a random direction as shown by the yellow nodes in the graph. As shown in Fig. 13, the building connectivity can be quite complex and there has been limited research work that deals with this aspect. The graph structure shows that these connections between rooms cannot possibly be captured by a grid based or maze environment.

Also, note that, the double sided arrows in the graph enable transitions back and forth between rooms. This makes the environment more complicated for the agent since the agent could just go back and forth between 'safe' rooms and get stuck in a loop and may never converge. This point makes pretraining even more indispensable.

Since, the proposed method is able to reduce the action space by a lot, the neural network doesn't need to be made too large. The network configuration is given in Table 4. Note that the addition layer does not require any trainable parameters. The neural network is trained using the Adam optimizer [56] with default hyperparameter settings and a learning rate $\eta = 0.001$ for 5000 episodes. The training was performed on the NVIDIA DGX-2. The optimal number of steps for evacuation in the UiA building graph is around ~ 2000.

TABLE IV: Network Configuration

Type	**Size**	**Activation**
Dense	512	ReLU
Dense	1024	ReLU
Dense	1024	ReLU
Dense	1024	ReLU
Dense	8281	Linear
Addition	-	-

C. *Results*

The results of our proposed method consisting of shortest path Q-matrix transfer learning to Dueling-DQN model with one-step simulation and action importance vector acting as an attention mechanism applied on the University of Agder's A,B,C and D blocks consisting of 91 rooms and 8281 actions (whose graph is shown in Fig. 13) is shown in Fig. 14. The performance numbers are given below:

- **Average number of time-steps:** 2234.5
- **Minimum number of time-steps:** ~2000
- **Training time (per episode):** 32.18 *s*

The graph in Fig. 14 shows the convergence of our method with evacuation time-steps on the y-axis and the episode number on the x-axis. It takes slightly longer to converge compared to the convergence in previous small example environments. This is obviously due to the size of the environment and complex connectivity. But overall the performance of our model is excellent.

After $\sim$ 1900 episodes, the algorithm has almost converged. There are a few spikes suggesting fluctuations from the optimal behaviour due to the dynamic nature of the environment and the uncertainty in actions. After $\sim$ 3300 episodes, the algorithm completely converges in the range $(2000 - 2070)$ times-steps for total evacuation. The method cannot converge to the minimum possible time-steps $= 2000$ because of the fire spread dynamics, encountering bottleneck conditions and action uncertainty.

The results clearly suggest that even though the proposed fire evacuation environment is dynamic, uncertain and full of constraints, our proposed method using novel action reduction technique with attention based mechanism and transfer learning of shortest path information is able to achieve excellent performance on a large and complex real world building model. This further confirms that, with a minute added overhead of one-step simulation and action importance vector, our method is scalable to much larger and complex building models.

VI. Conclusion

In this paper, we propose the first realistic fire evacuation environment to train reinforcement learning agents. The environment is implemented in OpenAI gym format. The environment has been developed to simulate realistic fire scenarios. It includes features like fire spread with the help of exponential decay reward functions and degree functions, bottlenecks, uncertainty in performing an action and a graph based environment for accurately mapping a building model. We also propose a new reinforcement learning method for training on our environment. We use tabular Q-learning to generate q-values for shortest path to the exit using the adjacency matrix of the graph based environment. Then, the result of Q-learning (after being offset by a σ) is used to pretrain the DQN network weights to incorporate shortest

[1]UiA building map can be found here: https://use.mazemap.com/#v=1&zlevel=2&left=8.5746533&right=8.5803711&top=58.3348318&bottom=58.3334208&campusid=225

C

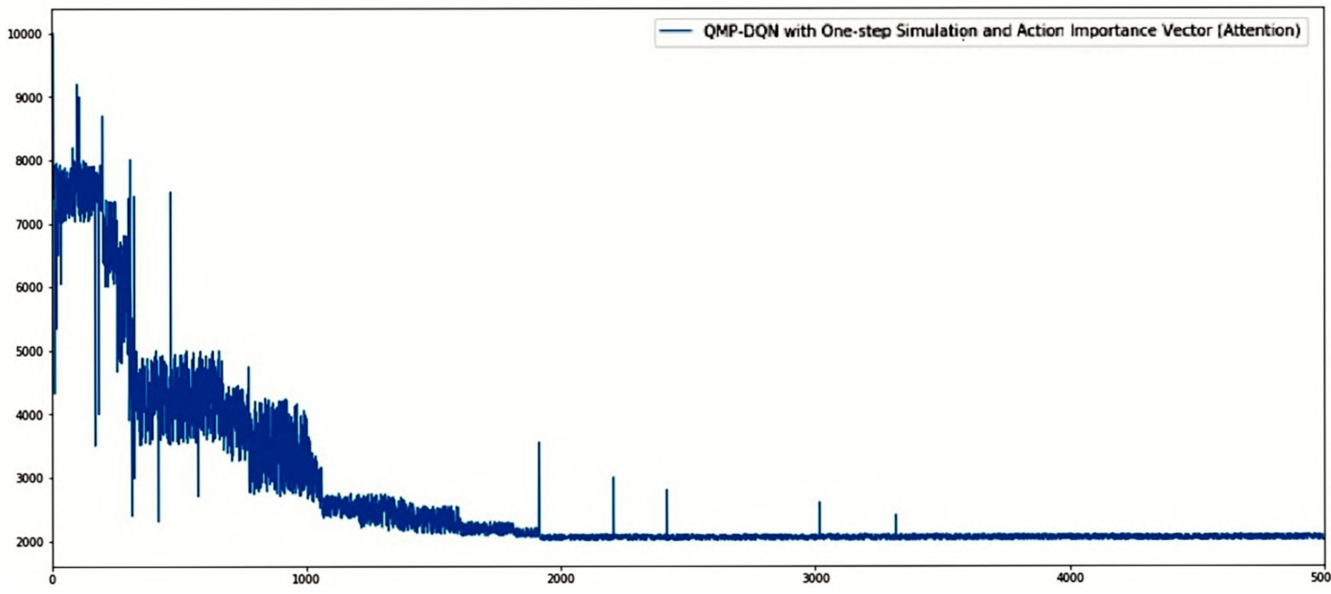


Fig. 14: Proposed method applied on the UiA Building

path information in the agent. Finally, the pretrained weights of the DQN based agents are trained on the fire evacuation environment.

We prove the faster convergence of our method using Task Transfer Q-learning theorems and the convergence of Q-learning for the shortest path task. The Q-matrix pretrained DQN agents (QMP-DQN) are compared with state-of-the-art reinforcement learning algorithms like DQN, DDQN, Dueling-DQN, PPO, VPG, A2C, ACKTR and SARSA on the fire evacuation environment. The proposed method is able to outperform all these models on our environment on the basis of convergence, training time and stability. Also, the comparisons of QMP-DQN with original DQN based models show clear improvements over the latter.

Finally, we show the scalability of our method by testing it on a real world large and complex building model. In order to reduce the large action space (8281 actions), we use the one-step simulation technique on the pretraining environment instance to calculate the action importance vector, which can be thought of as an attention based mechanism. The action importance vector gives the best k actions a weight of 0 and the rest are assigned a large negative weight of -9999 (to render the Q-values of these too low to be selected by the Q-function). This reduces the action space by $\sim 90\%$ and our proposed method, QMP-DQN model, is applied on this reduced action space. We test this method on the UiA, Campus Grimstad building, with the environment consisting of 91 rooms. The results show that this combination of methods works really well in a large real world fire evacuation emergency environment.

Acknowledgment

The authors would like to thank Tore Olsen, Chief of the Grimstad Fire Department, and the Grimstad Fire Department for supporting us with their expertise regarding fire emergencies and evacuation procedures as well as giving feedback to improve our proposed environment and evacuation system. We would also like to thank Dr. Jaziar Radianti, Center for Integrated Emergency Management (CIEM), University of Agder, for her input to this research work.

References

[1] Abbas Abdolmaleki, Mostafa Movahedi, Sajjad Salehi, Nuno Lau, and Luis Paulo Reis. A reinforcement learning based method for optimizing the process of decision making in fire brigade agents. In Luis Antunes and H. Sofia Pinto, editors, *Progress in Artificial Intelligence*, pages 340–351, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[2] Fatemeh Pahlevan Aghababa, Masaru Shimizu, Francesco Amigoni, Amirreza Kabiri, and Arnoud Visser. Robocup 2018 robocup rescue simulation league virtual robot competition rules document, May 2018.

[3] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report TR 166, Cambridge University Engineering Department, Cambridge, England, 1994.

[4] Daniel Moura and Eugenio Oliveira. Fighting fire with agents: An agent coordination model for simulated firefighting. In *Proceedings of the 2007 Spring Simulation Multiconference - Volume 2*, SpringSim '07, pages 71–78, San Diego, CA, USA, 2007. Society for Computer Simulation International.

[5] Haifeng Zhao and Stephan Winter. A time-aware routing map for indoor evacuation. *Sensors*, 16(1), 2016.

[6] Ashley Wharton. Simulation and investigation of multi-agent reinforcement learning for building evacuation scenarios *. 2009.

[7] Mei Ling Chu, Paolo Parigi, Kincho Law, and Jean-Claude Latombe. Modeling social behaviors in an evacuation simulator. *Computer Animation and Virtual Worlds*, 25(3-4):373–382.

[8] V. J. Cassol, E. Smania Testa, C. Rosito Jung, M. Usman, P. Faloutsos, G. Berseth, M. Kapadia, N. I. Badler, and S. Raupp Musse. Evaluating and optimizing evacuation plans for crowd egress. *IEEE Computer Graphics and Applications*, 37(4):60–71, 2017.

[9] LH Lee and Young-Jun Son. Dynamic learning in human decision behavior for evacuation scenarios under bdi framework. In *Proceedings of the 2009 INFORMS Simulation Society Research Workshop. INFORMS Simulation Society: Catonsville, MD*, pages 96–100, 2009.

[10] Seungho Lee, Young-Jun Son, and Judy Jin. An integrated human decision making model for evacuation scenarios under a bdi framework. *ACM Trans. Model. Comput. Simul.*, 20(4):23:1–23:24, November 2010.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[12] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627, 2018.

[13] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017.

[14] Pete Shinners. Pygame. http://pygame.org/, 2011.

[15] Łukasz Kidziński, Sharada P Mohanty, Carmichael Ong, Jennifer Hicks, Sean Francis, Sergey Levine, Marcel Salathé, and Scott Delp. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In Sergio Escalera and Markus Weimer, editors, *NIPS 2017 Competition Book*. Springer, Springer, 2018.

[16] Stephen Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644, Jul 1983.

[17] S. I. Pak and T. Hayakawa. Forest fire modeling using cellular automata and percolation threshold analysis. In *Proceedings of the 2011 American Control Conference*, pages 293–298, June 2011.

[18] Marco Wiering and Marco Dorigo. Learning to control forest fires. *Ultrech University Repository*, Jan 1998.

[19] G. E. Sakr, I. H. Elhajj, G. Mitri, and U. C. Wejinya. Artificial intelligence for forest fire prediction. In *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1311–1316, July 2010.

[20] Sriram Ganapathi Subramanian and Mark Crowley. Using spatial reinforcement learning to build forest wildfire dynamics models from satellite images. *Frontiers in ICT*, 5:6, 2018.

[21] Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[22] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *CoRR*, abs/1801.06146, 2018.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[24] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[25] Yue Wang, Qi Meng, Wei Chen, Yuting Liu, Zhiming Ma, and Tie-Yan Liu. Target transfer q-learning and its convergence analysis. *CoRR*, abs/1809.08923, 2018.

[26] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[27] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[28] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[29] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30 – 37. Morgan Kaufmann, San Francisco (CA), 1995.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[31] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[35] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[37] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, March 2016.

[38] Chung-Cheng Chiu, Tara Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Katya Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models. 2018.

[39] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772. JMLR Workshop and Conference Proceedings, 2014.

[40] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012.

[41] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[42] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, May 2013.

[43] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, Jan 2012.

[44] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA, 2015. MIT Press.

[45] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. *CoRR*, abs/1804.03235, 2018.

[46] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849, 2018.

[47] Lierni Sestorain, Massimiliano Ciaramita, Christian Buck, and Thomas Hofmann. Zero-shot dual machine translation. *CoRR*, abs/1805.10338, 2018.

[48] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567, 2018.

[49] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[50] H. D. Patino and D. Liu. Neural network-based model reference adaptive control system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 30(1):198–204, Feb 2000.

[51] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

[52] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[53] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[54] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[55] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[56] D.P. Kingma and L.J. Ba. Adam: A method for stochastic optimization. In *ICLR*, International Conference on Learning Representations (ICLR), page 13, San Diego, CA, USA, 7–9 May 2015. Ithaca, NY: arXiv.org.

[57] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Pittsburgh, PA, USA, 1992. UMI Order No. GAX93-22750.

[58] Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc., 2010.

[59] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2094–2100. AAAI Press, 2016.

[60] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1995–2003. JMLR.org, 2016.

[61] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[62] Csaba Szepesvári. The asymptotic convergence-rate of q-learning. In *Advances in Neural Information Processing Systems*, pages 1064–1070, 1998.

[63] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Comput.*, 6(6):1185–1201, November 1994.

[64] John N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202, Sep 1994.

[65] E. Larsson. Evaluation of pretraining methods for deep reinforcement learning. 2018.

[66] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[67] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[68] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudik, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[69] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[70] François Chollet et al. Keras. https://keras.io, 2015.

[71] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[72] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[73] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1928–1937. JMLR.org, 2016.

[74] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[75] Yuhuai Wu, Elman Mansimov, Shun Liao, Alec Radford, and John Schulman. Openai baselines: Acktr and a2c. https://openai.com/blog/baselines-acktr-a2c/, 2017.

[76] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger B. Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *CoRR*, abs/1708.05144, 2017.

[77] James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. *CoRR*, abs/1503.05671, 2015.

[78] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[79] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[80] Gabriel Dulac-Arnold, Richard Evans, Peter Sunehag, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *CoRR*, abs/1512.07679, 2015.

[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[82] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[83] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

Paper D

Title:	Emergency Analysis: Multitask Learning with Deep Convolutional Neural Networks for Fire Emergency Scene Parsing
Authors:	Jivitesh Sharma, Ole-Christoffer Granmo and Morten Goodwin
Affiliation:	Dept. of Information and Communication Technology, University of Agder (UiA), Grimstad, Norway
Conference:	34th *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)*, July. 2021, Kuala Lumpur, Malaysia.
Copyright ©:	Springer

Paper E

Title:	Environment Sound Classification using Multiple Feature Channels and Attention based Deep Convolutional Neural Network
Authors:	Jivitesh Sharma, Ole-Christoffer Granmo and Morten Goodwin
Affiliation:	Dept. of Information and Communication Technology, University of Agder (UiA), Grimstad, Norway
Conference:	21st *Conference of the International Speech Communication Association (INTERSPEECH)*, Oct. 2020, Shanghai, China.
Copyright ©:	International Speech Communication Association (ISCA)

Environment Sound Classification using Multiple Feature Channels and Attention based Deep Convolutional Neural Network

Jivitesh Sharma[1], *Ole-Christoffer Granmo*[1], *Morten Goodwin*[1]

[1]Centre for Artificial Intelligence Research
Department of Information and Communication Technology
University of Agder, Norway

jivitesh.sharma@uia.no, ole.granmo@uia.no, morten.goodwin@uia.no

Abstract

In this paper, we propose a model for the Environment Sound Classification Task (ESC) that consists of multiple feature channels given as input to a Deep Convolutional Neural Network (CNN) with Attention mechanism. The novelty of the paper lies in using multiple feature channels consisting of Mel-Frequency Cepstral Coefficients (MFCC), Gammatone Frequency Cepstral Coefficients (GFCC), the Constant Q-transform (CQT) and Chromagram. And, we employ a deeper CNN (DCNN) compared to previous models, consisting of spatially separable convolutions working on time and feature domain separately. Alongside, we use attention modules that perform channel and spatial attention together. We use the mix-up data augmentation technique to further boost performance. Our model is able to achieve state-of-the-art performance on three benchmark environment sound classification datasets, i.e. the UrbanSound8K (97.52%), ESC-10 (94.75%) and ESC-50 (87.45%).

Index Terms: Convolutional Neural Networks, Attention, Multiple Feature Channels, Environment Sound Classification

1. Introduction

One of the most important application is the Environment Sound Classification (ESC) that deals with distinguishing between sounds from the real environment. It is a complex task that involves classifying a sound event into an appropriate class such as siren, dog barking, airplane, people talking etc.

The most successful ESC models consist of one or more standard audio feature extraction techniques and deep neural networks. In this paper, we explore the idea of employing multiple feature extraction techniques like the Mel-frequency Cepstral Coefficients (MFCC) [1], Gammatone Frequency Cepstral Coefficients (GFCC) [2], Constant Q-Transform (CQT) [3], Chromagram [4] and stack them to create a multiple channel input to our classifier.

After feature extraction, the next stage is classification. Many machine learning algorithms have been used to classify sound, music or audio events. However, in the ESC task, Deep CNNs have been able to outperform other techniques, as evident from the previous ESC models. In this paper, we also employ a Deep CNN for classification. However, we split between time and frequency domain feature processing by using separable convolutions [5] with different kernel sizes. Also, we use max pooling across only one of the domains at a time, until after the last set of convolutional layers to combine time and frequency domain features. This enables processing time and frequency domain features separately and then combining them at a later stage.

Along with the model, we also design a novel attention module that enables both spatial and channel attention. In order to achieve both spatial and channel attention with the same module, we need an attention weight matrix with dimensions equal to the DCNN block output. So that, each output feature map in each channel has it's own attention weights. We use the depthwise separable convolution [6] to achieve attention with minimal increase in number of parameters.

Using these techniques allows our model to achieve state-of-the-art performance on three benchmark datasets for environment sound classification task, namely, ESC-10, ESC-50 [7] and UrbanSound8K [8].

2. Related Work

There have been several innovative and high performance approaches proposed for the task of environmental sound classification (ESC). In [9], a deep CNN was shown to give competitive results for the ESC tasks by thorough and exhaustive experimentation on the three benchmark datasets.

In [10], phase encoded filterbank energies (PEFBEs) was proposed as a novel feature extraction technique. Finally, a score-level fusion of FBEs and PEFBEs with a CNN classifier achieved best performance.

In the second version of the EnvNet, called EnvNetv2 [11], the authors employed a mechanism called Between Class (BC) learning. In BC learning, two audio signals from different classes are mixed with each other with a random ratio. The CNN model is then fed the mixed sound as input and trained to output this mixing ratio.

An unsupervised approach of learning a filterbank from raw audio signals was proposed in [12]. Convolutional Restricted Boltzmann Machine (ConvRBM), which is an unsupervised generative model, was trained to raw audio waveforms. A CNN is used as a classifier along with ConvRBM filterbank and score-level fusion with Mel filterbank energies. Their model achieves 86.5% on the ESC-50 dataset.

A novel data augmentation technique, called mixup, was proposed in [13]. It consists of mixing two audio signals and their labels, in a linear interpolation manner, where the mixing is controlled by a factor λ. In this way, their model achieves 83.7% accuracy on the UrbanSound8K dataset. We employ the mix-up data augmentation in our work to boost our model's performance.

A complex two stream structure deep CNN model was proposed in [14]. It consists of two CNN streams. One is the LMCNet which works on the log-mel spectrogram, chroma, spectral contrast and tonnetz features of audio signals and the other is the MCNet which takes MFCC, chroma, spectral contrast and tonnetz features as inputs. The decisions of the two CNNs are fused to get the final TSDCNN-DS model. It achieves 97.2% accuracy on the UrbanSound8K dataset.

There have also been a few contributions towards the ESC task

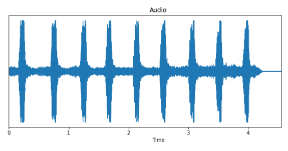
that consist of attention based systems. In [15], a combination of two attention mechanisms, channel and temporal, was proposed. The temporal attention consists of 1×1 convolution for feature aggregation followed by a small CNN to produce temporal attention weights. On the other hand, channel attention consists of a bank of fully connected layers to produce the channel attention map. Using two separate attention models makes the system very complex and increases the number of parameters by a lot. We perform spatial and channel attention with just one depthwise convolutional layer.
A multi-stream network with temporal attention for the ESC task was proposed in [16]. The model consists of three streams with each stream receiving one of the three stacked inputs: raw waveform, STFT (Short-time Fourier Transform) and delta STFT. A temporal attention model received the inputs directly and propagated it's output to the main models intermediate layers. Here, again, the model is too complex and also, the attention block doesn't receive any intermediate feedback from the main model.
The research works mentioned above and many others provide us with many insights by achieving high performance on difficult datasets. But, they also suffer from issues regarding feature extraction, computational complexity and CNN model architecture. In this paper, we try to address these issues and in doing so, achieve state-of-the-art performance.

3. Proposed Environment Sound Classification Model

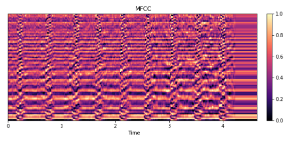
We propose a novel ESC model that consists of multiple feature channels extracted from the audio signal and a new DCNN architecture consisting of separable convolutions, that works on time and frequency domain separately and a depthwise convolution based attention mechanism.
The feature extraction stage consists of four channels of features, which are: Mel-Frequency Cepstral Coefficients (MFCC), Gammatone Frequency Cepstral Coefficients (GFCC), Constant Q-transform (CQT) and Chromagram.
For the classification stage, we propose a CNN architecture that works better for audio data, as shown in Fig. 3. We use spatially separable convolutions to process time and frequency domain features separately and aggregate them at the end. Also, the downsampling value is different for time and frequency domains in the maxpooling layers. Along side the main DCNN model, we add spatial and channel attention using the depthwise convolution. In the subsequent sub-sections, we explain the feature extraction and classification stages of our model.

3.1. Multiple Feature Channels

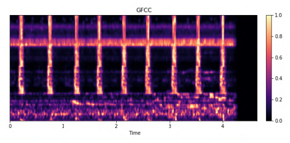
In this paper, we employ four major audio feature extraction techniques to create a four channel input for the Deep CNN, namely, Mel-Frequency Cepstral Coefficients (MFCC) [1], Gammatone Frequency Cepstral Coefficients (GFCC) [2], Constant Q-Transform [3] and Chromagram [4]. Incorporating different signal processing techniques that extract different types of information provides the CNN with more distinguishable characteristics and complementary feature representations to accurately classify audio signals.
The MFCC, GFCC, CQT and Chroma features are stacked together to create a four channel input for the Deep CNN. Each feature plays it's part in the classification task. MFCC acts as the backbone by providing rich features, GFCC adds transient sound features, CQT contributes with better low-to-mid frequency range features and finally Chromagram provides pitch category analysis and signal structure information. Fig. 1 shows a graphical representation of the features extracted from an audio signal (Fig. 1(a)). All features are normalized between 0 and 1 using min-max normalization. From the figure, we can see the contrast in the values of each feature.



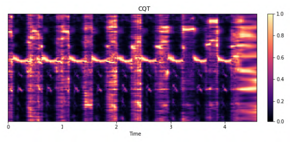
(a) *Audio Signal*



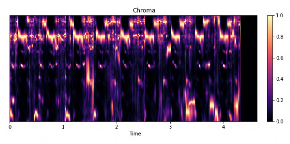
(b) *Mel Frequency Cepstral Coefficients*



(c) *Gammatone Frequency Cepstral Coefficients*

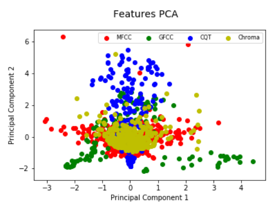


(d) *Constant Q-Transform*

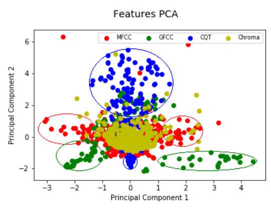


(e) *Chromagram*

Figure 1: *Multiple Feature Channels*



(a)



(b)

Figure 2: *PCA of Features*

Fig. 2(a) shows the Principal Component Analysis (PCA) of the features. We take the first two principal components of the four features we use in our model to create a 2D visualization of the feature space. From the figure we can see that most of features are heavily concentrated in the middle region. But, as shown in Fig. 2(b), we encircle a few regions that different features provide some amount of different information. Indeed some of these regions might contain irrelevant or outlier information that is not of value to classification. But, as seen in the figure these feature extraction techniques do provide unique and complementary information. Chromagram features provide little distinctive information and shown in the results section, it provides little increase to the performance of the model.

3.2. Deep CNN Architecture: Main Block

Fig. 3 shows our proposed Deep CNN architecture for environmental sound classification. The main block consists of five repetitions of *Conv2D-Conv2D-Conv2D-MaxPool-BatchNorm* with different number of kernels and kernel sizes. Almost all convolutional layers are made up of spatially separable convolutions.

In the case of the ESC task, the input are the features extracted from the audio signals. Each feature set is of the shape (t, f, c), where t is the compressed time domain (compressed due to window size and hop length) and c is the number of channels. Each window of time yields f number of features ($f = 128$ in our model). So, we treat the time domain and the feature domain separately. The kernels with the form $1 \times m$ work on the feature domain and the ones with $n \times 1$ work on the time domain. Using the $1 \times m$ type of convolution operation enables the network to process each set of features from a time window separately. And, the $n \times 1$ type of convolution allows the aggregation of a feature along the time domain. Now, c corresponds to the number of feature extraction methods we adopt (in our model, $c = 4$). So, each kernel works on each channel, which means that all different types of features extracted from the signal feature extraction techniques is aggregated by every kernel. Each kernel can extract different information from an aggregated combination of different feature sets.

Another major advantage of using this type of convolution is the reduction in number of parameters. This is the primary advantage of separable convolutions when they were used in [5] and have probably been used earlier as well.

In case of standard square kernels like $n \times n$, which are used for computer vision tasks, the dimensions of the kernel are in accordance to the image's spatial structure. The 2D structure of an image represents pixels, i.e. both dimensions of an image represent the same homogeneous information. Whereas, in case of audio features, one dimension gives a compact representation of frequency features of a time window and the other dimension represents the flow of time (or sliding time window). So, in order to process information accordingly and respect the information from different dimensions of the input, we use $1 \times m$ and $n \times 1$ separable convolutions.

3.3. Deep CNN Architecture: Attention Block

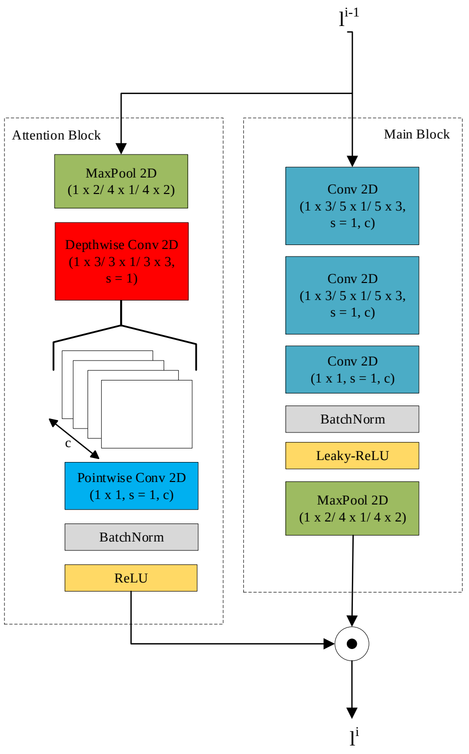
In this paper, we achieve spatial and channel wise attention using a single attention module and dramatically reduce the number of parameters required for attention by using depthwise convolutions. The attention block, shown in Fig. 3, runs in parallel with a main block. The pooling size and kernel size in the attention block is the same as the pooling and kernel size in the corresponding parallel main block. Using depthwise convolution reduces the number of parameters and thus reduces the overhead of adding attention blocks to the model.

Before the element-wise multiplication of the attention matrix with the main block output, we add a batch normalization layer to normalize the attention weights. Normalization is important for smoothing. The batch-norm layer is followed by a ReLU activation, that makes the attention weight matrix sparse which makes the element-wise multiplication computationally efficient.

$$a^i = \phi(BatchNorm(f(MaxPool(l^{i-1})))) \tag{1}$$

$$l^i = a^i \odot \hat{l^i} \tag{2}$$

Equations 3 and 4 make up the attention module, where f is the depthwise separable convolution comprising of depthwise and point-wise convolution and ϕ is the ReLU activation function. This single attention module performs both spatial and channel attention. Channel-wise attention requires an attention



l^{i-1}
Attention Block
Main Block
MaxPool 2D (1 x 2/ 4 x 1/ 4 x 2)
Depthwise Conv 2D (1 x 3/ 3 x 1/ 3 x 3, s = 1)
c
Pointwise Conv 2D (1 x 1, s = 1, c)
BatchNorm
ReLU
Conv 2D (1 x 3/ 5 x 1/ 5 x 3, s = 1, c)
Conv 2D (1 x 3/ 5 x 1/ 5 x 3, s = 1, c)
Conv 2D (1 x 1, s = 1, c)
BatchNorm
Leaky-ReLU
MaxPool 2D (1 x 2/ 4 x 1/ 4 x 2)
l^i

Figure 3: *Attention based DCNN model*

weight for each output channel of the main block and spatial attention requires an attention weight for each spatial location in the output feature map. Our attention module produces c weights, which enables channel attention, and each weight in c is a matrix of $n \times m$, which enables spatial attention. And, using a single depthwise separable convolution layer we are able to achieve this with considerably less number of parameters and operations.

An advantage of using attention as a separate module that runs in parallel with every main block and connected before and after each main block, with less number of parameters and layers, is that it allows smooth propagation of the gradient like skip or residual connections [17, 18].

4. Experimental Setup

We report state-of-the-art results on ESC benchmark datasets, i.e. UrbanSound8K, ESC-10 and ESC-50, using the proposed model. The ESC-10 and ESC-50 contain 2000 audio files of 5 seconds length each, while UrbanSound8K consists of 8732 audio files of 4 seconds each. ESC-10 and UrbanSound8K contain audio from 10 classes while ESC-50 has 50 classes. We use k-fold cross-validation on the specified folds and report the average accuracy across the folds. For ESC-10 and ESC-50, $k = 5$ and for UrbanSound8K, $k = 10$.

We use Tensorflow and Keras to implement our CNN classifier and Librosa [19] and the Matlab Signal Processing Toolbox [20] for audio processing and feature extraction. In terms of hardware, we use the NVIDIA DGX-2 consisting of 16 NVIDIA Tesla $V100$ GPUs with 32 Gigabytes of VRAM each and a system memory of 1.5 Terabytes.

For every feature extraction technique, we extract 128 features for each window of length 1024 (3.2 ms) with a hop length of 512 (1.6 ms) at 32kHz. We normalize all feature vectors using min-max normalization. Our DCNN model is trained to minimize the categorical cross-entropy loss using the Adam optimizer with Nestorov momentum, along with Dropout of 0.25 after the dense layer. and weight decay of $\lambda = 0.1$. We run our model for 500 epochs per fold. We set the initial learning rate of training to 0.01 and decrease it by a factor of 10 every 150 epochs.

As shown in [13], mix-up data augmentation plays a very important role in improving performance, especially when the model is large and data is scarce. We use a mini-batch size of 200. Table 1 displays the results of previous state-of-the-art ESC models that tested their methods on one or more of the three benchmark datasets. All of these models have been briefly described in Section 2. The last row of the table shows the results of our proposed model on the three datasets.

Table 1: *Previous state-of-the-art ESC models vs Proposed model*

Model	ESC-10	ESC-50	US8K
EnvNetv2+strong augment [11]	91.30	84.70	78.30
PiczakCNN [9]	90.20	64.50	73.70
CNN+Mixup [13]	91.70	83.90	83.70
FBEs⊕ConvRBM-BANK [12]	-	86.50	-
CRNN+channel & temporal Attention [15]	94.20	86.50	-
Multi-stream+temporal Attention [16]	94.20	84.00	-
TSCNN-DS [14]	-	-	97.20
Multiple Feature Channel + Deep CNN with Attention (Proposed)	**94.75**	**87.45**	**97.52**

5. Results

We show the advantages of using multiple features, data augmentation, depthwise convolutions and attention mechanism from our experiments on the three benchmark datasets[1]. Using separable convolutions (spatial or depthwise), has the advantage of reducing the number of parameters in the model. We use spatially separable convolutions in our main block and depthwise separable convolutions in the attention block. In Table 2, we show the effect of using separable convolutions in terms of the number of parameters and model performance. The DCNN-5 is the model without attention and DCNN-5 SC is with standard convolutions instead of separable convolutions. The separable convolutions, 1×3 and 5×1, is replaced by 5×3 convolution operation. We use padding when necessary to keep the model depth valid according to the input, since standard rectangular convolutions reduce the output dimensions more quickly.

From Table 2, we can see that, for the task of environment sound classification, the spatially separable convolutions have less number of parameters and perform better than standard convolutions. DCNN-5 SC has 130K more parameters than DCNN-5 and obtains 3.25% lower accuracy than DCNN-5 on the ESC-50. Adding the attention mechanism just adds 20K more parameters and increases the performance by 2.7%, courtesy of depthwise convolutions. Using standard convolutions to build the attention model results in an increase of 90K parameters and 0.4% accuracy.

These findings are consistent with the UrbanSound8K dataset. The difference in the number of parameters between the datasets for the same models is because of the difference in input shapes. UrbanSound8K has 4 seconds long audio files, whereas, ESC-50 has 5 seconds long. So, both of them sampled at 32kHz produce different number of time windows. The input shape for ESC-50 is $\langle 313, 128, 4 \rangle$ and for UrbanSound8K is $\langle 250, 128, 4 \rangle$ represented as $\langle$time-windows, features, channels$\rangle$. We also test our model with fewer number of features extracted by the audio feature extraction methods. Table 3 shows the results when the number of features are reduced. The model accuracy monotonically increases with the increase in the number of features. We stop at 128 features, which produces the best results, to avoid increasing the complexity of the model.

The same tests were conducted on the ESC-10 dataset. The results were consistent with the findings shown above. ESC-10 is a subset of the ESC-50 dataset. We also report state-of-the-art performance on the ESC-10 dataset with 94.75% accuracy.

[1]The Table containing the results of our experiments with different combination of features and the effect of data augmentation is attached as supplementary material, due to lack of space

Table 2: *Performance Comparison of Number of Parameters on ESC-50 and UrbanSound8K*

Model	Parameters ESC-50	ESC-50	Parameters US8K	US8K
DCNN-5	1.27M	84.75	0.87M	94.25
ADCNN-5	**1.29M**	**87.45**	**0.89M**	**97.52**
DCNN-5 SC	1.40M	81.50	1.04M	91.25
ADCNN-5 (without Depthwise Sep. Conv.)	1.36M	87.05	0.97M	96.35

Table 3: *Performance of different number of feature coefficients on ESC-50 and UrbanSound8K*

Model	# Features	ESC-50	US8K
ADCNN-5	48	80.12	89.25
	64	85.25	94.25
	96	86.15	95.50
	128	**87.45**	**97.52**

6. Conclusions

We propose a novel approach for environmental sound classification that consists of multiple feature channels and attention based deep convolutional neural network with domain wise convolutions. We combine feature extraction methods like the MFCC, GFCC, CQT and Chromagram to create a multi channel input for the CNN classifier. The model consists of two block: Main block and Attention block. We employ a Deep CNN consisting of separable convolutions in the main block. The separable convolutions work on the time and feature domains separately. Parallel to the main blocks, we also use an attention mechanism that consists of depthwise separable convolution. Both channel and spatial attention are achieved using a small increase in number of parameters. We test our model on the three benchmark datasets: ESC-10, ESC-50 and UrbanSound8K. We use mix-up data augmentation techniques to further improve performance. Our model achieves 94.75%, 87.45% and 97.52% accuracy on ESC-10, ESC-50 and UrbanSound8K respectively, which is state-of-the-art performance on all three datasets.

7. References

[1] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, August 1980.

[2] Y. Shao, Z. Jin, D. Wang, and S. Srinivasan, “An auditory-based feature for robust speech recognition,” in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 4625–4628.

[3] C. Schörkhuber, “Constant-q transform toolbox for music processing,” 2010.

[4] R. N. Shepard, “Circularity in judgments of relative pitch,” *The Journal of the Acoustical Society of America*, vol. 36, no. 12, pp. 2346–2353, 1964.

[5] F. Mamalet and C. Garcia, “Simplifying convnets for fast learning,” in *Artificial Neural Networks and Machine Learning – ICANN 2012*, A. E. P. Villa, W. Duch, P. Érdi, F. Masulli, and G. Palm, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 58–65.

[6] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[7] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[8] J. Salamon, C. Jacoby, and J. P. Bello, “A dataset and taxonomy for urban sound research,” in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM ’14. New York, NY, USA: ACM, 2014, pp. 1041–1044. [Online]. Available: http://doi.acm.org/10.1145/2647868.2655045

[9] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sep. 2015, pp. 1–6.

[10] R. N. Tak, D. M. Agrawal, and H. A. Patil, “Novel phase encoded mel filterbank energies for environmental sound classification,” in *Pattern Recognition and Machine Intelligence*, B. U. Shankar, K. Ghosh, D. P. Mandal, S. S. Ray, D. Zhang, and S. K. Pal, Eds. Cham: Springer International Publishing, 2017, pp. 317–325.

[11] Y. Tokozume, Y. Ushiku, and T. Harada, “Learning from between-class examples for deep sound recognition,” *CoRR*, vol. abs/1711.10282, 2017. [Online]. Available: http://arxiv.org/abs/1711.10282

[12] J. Salamon and J. P. Bello, “Unsupervised feature learning for urban sound classification,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 171–175.

[13] Z. Zhang, S. Xu, S. Cao, and S. Zhang, “Deep convolutional neural network with mixup for environmental sound classification,” in *Pattern Recognition and Computer Vision*, J.-H. Lai, C.-L. Liu, X. Chen, J. Zhou, T. Tan, N. Zheng, and H. Zha, Eds. Cham: Springer International Publishing, 2018, pp. 356–367.

[14] Y. Su, K. Zhang, J. Wang, and K. Madani, “Environment sound classification using a two-stream cnn based on decision-level fusion,” *Sensors*, vol. 19, no. 7, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1733

[15] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, “Learning attentive representations for environmental sound classification,” *IEEE Access*, vol. 7, pp. 130 327–130 339, 2019.

[16] X. Li, V. Chebiyyam, and K. Kirchhoff, “Multi-stream network with temporal attention for environmental sound classification,” *CoRR*, vol. abs/1901.08608, 2019. [Online]. Available: http://arxiv.org/abs/1901.08608

[17] M. S. Ebrahimi and H. K. Abadi, “Study of residual networks for image recognition,” *CoRR*, vol. abs/1805.00325, 2018. [Online]. Available: http://arxiv.org/abs/1805.00325

[18] A. E. Orhan, “Skip connections as effective symmetry-breaking,” *CoRR*, vol. abs/1701.09175, 2017. [Online]. Available: http://arxiv.org/abs/1701.09175

[19] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” 2015.

[20] *MATLAB Signal Processing Toolbox 2019*. Natick, Massachusetts, United States: The MathWorks Inc., 2019.

Paper F

Title:	Emergency detection with Environment Sound using Deep Convolutional Neural Networks
Authors:	Jivitesh Sharma, Ole-Christoffer Granmo and Morten Goodwin
Affiliation:	Dept. of Information and Communication Technology, University of Agder (UiA), Grimstad, Norway
Conference:	5th *International Congress on Information and Communication Technology (ICICT)*, Feb. 2020, London, UK.
Copyright ©:	Springer