UiA University of Agder

Master

An Attention-Based Encoder-Decoder for Week-Ahead Sales Forecasts Applied to a Norwegian Zoo

OLAV MARKUS SJURSØ

SUPERVISOR
DR. MORTEN GOODWIN

University of Agder, 2021
Faculty of Engineering and Science
Department of Engineering Sciences

Acknowledgements

This thesis concludes my studies in the Information and Communication Technology master program and five years of study at the University of Agder. I would first like to thank my supervisor, Dr. Morten Goodwin, for taking time out of his day each week to guide my thesis. His valuable feedback and discussions have helped me present my work in a structured manner.

I would also like to thank Dyreparken for giving me an exciting problem to work on and Daniel Sjøstrøm for providing expert domain knowledge.

In addition, I would like to thank the Norwegian Educational Loan Fund, without whose support this thesis would not be possible.

Abstract

Accurate sales forecasts are vital for companies to plan efficiently and make strategic economic decisions. However, predicting future sales is a complex problem due to the high variance and strong seasonality of sales data. Dyreparken is a zoo and water park in Norway that operates over 30 stores and restaurants throughout the two parks. Accurate automated sales forecasts can enable Dyreparken to more efficiently manage staffing by providing a more reliable decision basis. Such forecasting is challenging among others because there are significant differences between departments with varying activity levels, opening hours, and the uncertainty created by the weather.

This thesis investigates how we can apply machine learning-based sales forecasting to amusement parks and zoo's to forecast hourly turnover for the coming week. By structuring the problem as a multi-output prediction problem, we propose an attention-based encoder-decoder model with gated recurrent units. The proposed model is compared to a range of baseline models. Experimental results show that the suggested model outperforms models such as XGBoost, vanilla artificial neural networks, and neural networks with long short-term memory. Adding weather-related features to the model enables it to predict large deviations in sales caused by poor weather, which can help Dyreparken prevent large deficits in sales generated by unexpectedly low visitor numbers due to rain. In our best-case scenario, our proposed model can predict future turnover with a mean absolute error of 0.135, or 0.452 weighted MAPE.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Dyreparken is a zoo and amusement park located in southern Norway, consisting of two separate parks, Dyreparken (zoo) and Badelandet (water park). In 2019, Dyreparken hosted over one million guests [1] in both parks. For zoos, it is necessary to have a realistic forecast of future turnover to manage employees efficiently. Dyreparken currently uses manually created budgets. These provide a good approximation of future turnover but fail to consider short-term variables such as the weather. Having a more accurate prediction of turnover the week before allows Dyreparken and other destinations in this sector to manage employees more efficiently and move employees where they are needed. Furthermore, an accurate sales forecast can help mitigate risks associated with the weather.

Sales forecasting is a complex problem because of the high variance present in sales data. This is made even more complicated in Zoos because of strong seasonality, different opening hours, and varying activity levels. Another factor is that weather and other local effects can have a high impact on turnover because the number of visitors is highly dependent on the weather. Traditionally, sales forecasting problems have been solved with methods such as ARIMA [2], SVM [3], and vanilla neural networks [4]. Recently, deep neural networks have shown great promise in time series forecasting problems. This thesis takes a deep learning approach to the problem of sales forecasting in amusement parks and zoos.

1.1 Thesis Definition

Dyreparken currently uses manually created budgets which fails to account for short-term variables. This thesis aims to investigate the use of state-of-the-art deep-learning methods for short-term sales forecasting in amusement parks and zoos.

1.1.1 Thesis Goals

Goal 1: *Assemble a dataset suitable for multivariate time series forecasting.*

Goal 2: *Develop one or more models which can predict future sales.*

1.1.2 Hypotheses

Hypothesis 1: *Using past weather forecasts can increase the accuracy of sales forecasts in a Zoo.*

Hypothesis 2: *A model trained to only forecast the Summer break can yield more accurate predictions on summer forecasts than a model trained to forecast the entire year.*

Hypothesis 3: *It is possible to successfully pre-train a model with data from one location to increase the accuracy of revenue predictions of another.*

1.2 Previous Work

In IKT442, Sjursø [5] carried out a project on sales forecasting for Dyreparken. The project aimed to predict daily turnover for the next year. As part of one of the experiments, a model trained on all departments was compared to models trained on individual departments. The results showed a significant decrease in error when forecasts were generated with models trained on individual departments.

For this thesis, we make the assumption that the departments are so different that they cannot all be grouped to train one model.

1.3 Contributions

The main contribution of this thesis relates to the application of recent advances in time series forecasting to the problem of sales forecasting in zoos and amusement parks. An attention-based encoder-decoder model with gated recurrent units is used to forecasts hourly sales for the next week. This method has never been applied to a zoo or water park to the best of the author's knowledge. This problem is unique since it is a multi-site

time series forecasting problem where future revenue is dependent on the weather and the number of visitors.

1.4 Thesis Structure

The thesis is dividied into the following chapters:

- **Chapter 2** gives theoretical knowledge related to time series forecasting, methods of time series forecasting, and artificial intelligence. The last section outlines state-of-the-art research in the field of time series forecasting and explainable artificial intelligence.
- **Chapter 3** outlines the key methods used in this thesis.
- **Chapter 4** focuses on analyzing available data from Dyreparken and feature selection.
- **Chapter 5** reveals and discusses results from experiments.
- **Chapter 6** concludes the thesis and discusses future work.

Chapter 2

Background

This chapter starts by exploring theoretical background knowledge related to time series and common methods. The last part discusses deep learning methods for time series forecasting and explainable artificial intelligence.

2.1 Theory

2.1.1 Time Series Forecasting

Time series forecasting relates to prediction problems that involve a time component. Usual prediction problems pay no respect to time, but in time series forecasting, the prediction is dependent on time. Time Series models are trained on historical data with the goal of predicting future observations.

There are different types of time series forecasting problems. Univariate time series forecasting only relies on past observations when making predictions. In terms of machine learning, a model can be trained to predict observation y_t given the past two time step observations y_{t-1} and y_{t-2}. A multi-variate forecasting problem has multiple observations for each time step that relates to the target value. A multi-site time series forecasting problem is where multiple different locations are predicted.

Forecasting multiple future values is called multi-step forecasting. There are several different methods to achieve this. A direct approach would involve developing multiple models, one for each time step. Suppose the future value relies on the value of the previous time step. In that case, the predicted value of the previous time-step can be used as input for the next time step, predicting multiple steps in a recurrent fashion. Another possible approach uses multi-output prediction, where the model's output is a sequence of future time steps.

Sales forecasting is a specific time series forecasting problem where the aim is to predict future sales revenue or product sales. Accurate revenue forecasts allow a business to make better decisions related to goals, budgeting, employee rosters, and hiring. Sales forecasting can be handled as a time series forecasting problem, where historical data and past observations related to sales can be used to predict future revenue.

2.1.2 Recurrent Neural Networks

Traditional neural networks are unable to recollect past experiences. Recurrent Neural Networks (RNN) addresses this issue by adding loops, allowing information to persist in the network. Figure 2.1 shows a unit of a RNN. The network, A, takes some input X_t and outputs h_t. The loop in the network allows information to be passed from one step of the network to the next.

However, RNN has some shortcomings. It suffers from the vanishing gradient problem. The gradient is the value used to update the weights of the network. The vanishing gradient problem happens when the gradient shrinks as it backpropagates through time. If the gradient gets too small, it does not contribute to learning. Because of this, RNNs particularly struggle with temporal contingencies in the input/output that span long intervals [6]. i.e., it struggles to learn long-term dependencies in the data.

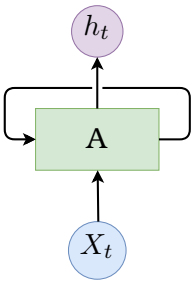

Figure 2.1: A Recurrent Neural Network unit.

2.1.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a variant of RNN capable of learning long-term dependencies. First introduced by Hochretier & Schmidhuber [7]. There is also a Bidirectional variant, where the network is trained on the original and a reversed copy of the input sequence [8]. LSTM has been utilized for many different tasks, such as handwriting recognition [9], medical imaging [10] and power load forecasting [11].

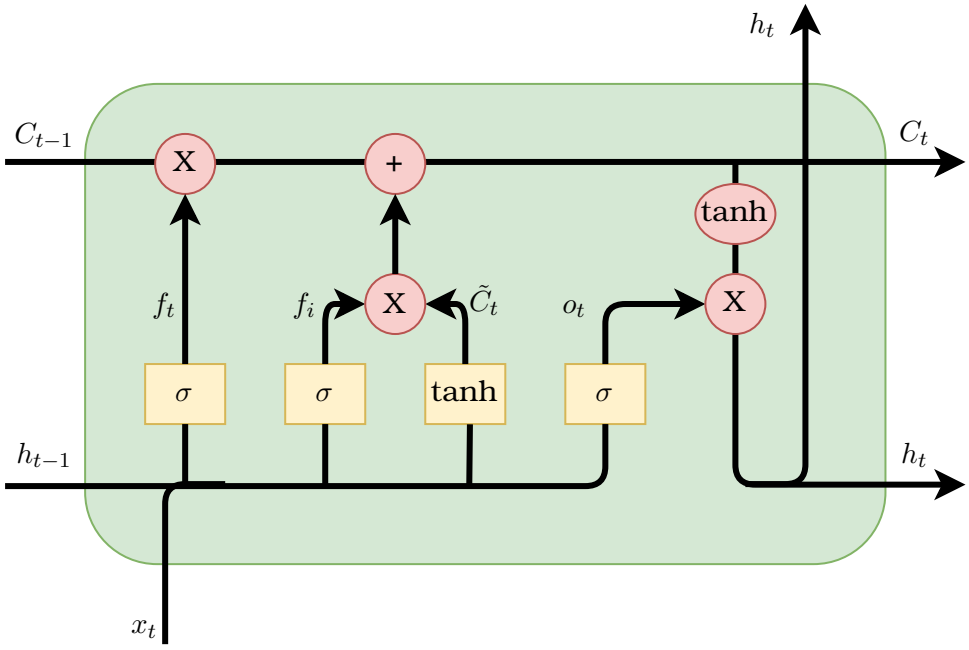

Figure 2.2: A Long Short Term Memory unit.

LSTM consists of two states and various gates as shown in figure 2.2. The cell state C_t acts as the memory of the network, passing relevant information from one time step to the next. The hidden state h_t contains information on previous inputs and is used for predictions. The forget gate enables information to be removed from the cell state, and the input gate enables the cell state to be updated with new information. The gates are neural networks with different activation functions. The gates have the ability to learn what information is relevant to keep or forget during training.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

The sigmoid activated forget gate layer decides what information should be kept or discarded from the cell state. Equation 2.1 shows the calculation done by the forget layer. The gate passes the previous hidden state h_{t-1} and the current input x_t through a sigmoid function which outputs a number between 0 and 1. A value closer to 1 means to keep, and a value closer to 1 means to forget. The number is then multiplied with the cell state. This is done for each value in the cell state c_{t-1}.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{2.2}$$

The next layer is called the Input Gate Layer. Together with a tanh activated layer, this layer decides what information to add to the cell state from the current time step. The previous hidden state and the current input are first passed to the sigmoid activation layer to calculate i_t, which decides what values will be updated. Next, the previous hidden state and current input are also passed to the tanh activated layer, which outputs a vector of candidate values $\tilde{C}_t$ scaled between -1 and 1. Then the output of the than layer $\tilde{C}_t$ is multiplied with the sigmoid output i_t, which decides what information to keep from the tanh layer.

$$C_t = f_t * c_{t-1} + i_t * \tilde{C}_t \tag{2.3}$$

The new cell state is given by equation 2.3. The previous cell state is multiplied with f_t in order to decide what is kept from the previous hidden state C_{t-1}. Next, this new state is concatenated with the new additions to the cell state.

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{2.4}$$

Lastly, the output gate calculates the new hidden state h_t as shown in equation 2.4. The hidden state is passed through a sigmoid layer in order to decide what information to keep. The cell state is run through a tanh layer and multiplied with the output of the sigmoid layer. The new hidden state is outputted and passed to the next time step together with the new cell state.

2.1.4 Gated Recurrent Unit

More recent modifications of LSTM have been proposed, such as Gated Recurrent Unit (GRU) [12]. GRU fuses the LSTM input and forget gate into a single update gate. Instead of outputting both a context vector and a hidden state, GRU encapsulates the information in a single vector h_t, which is outputted. The vector h_t acts as both the output of the unit

and the input to the next unit in the layer. Figure 2.3 shows a visualization GRU with its various gates, inputs, and outputs.

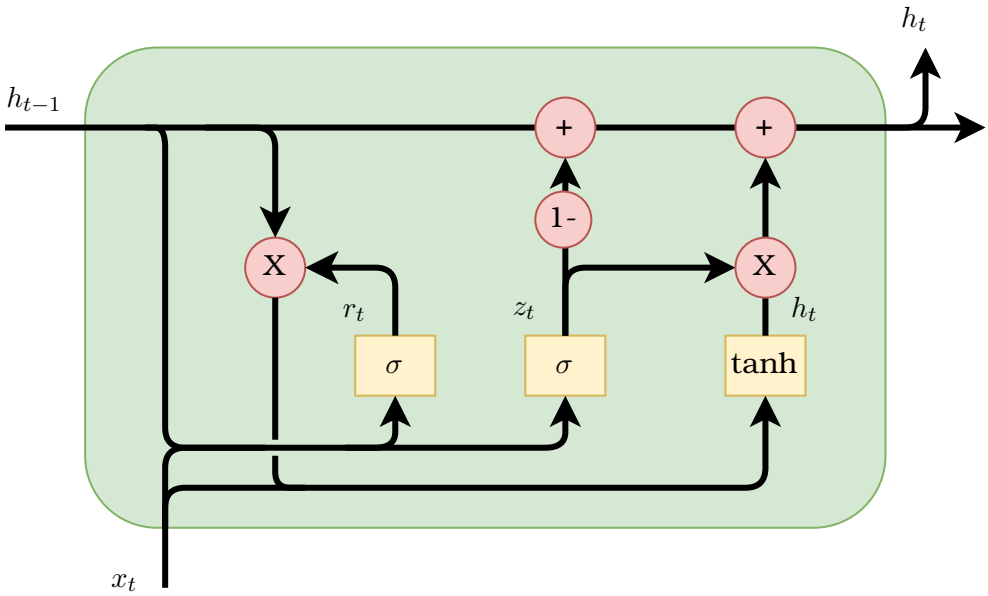

Figure 2.3: A Gated Recurrent Unit

First, we have the reset gate, which decides how much past information to keep. The calculation is shown in equation 2.5. First, the previous hidden state h_{t-1} is concatenated with the current input x_t. Next, this vector is multiplied element-wise with a weight W_z. Finally, this is run through a Sigmoid layer to produce the output r_t, which squishes the output between 0 and 1. A value of 0 would completely remove the previous state, while a value of 1 would indicate that the entire previous state should be kept.

$$r_t = \sigma(w_r \cdot [h_{t-1}, x_t] \tag{2.5}$$

The next gate is the update gate, which decides what new information to add and what information to remove. The update gate is computed as shown in equation 2.6.

$$Z_t = \sigma(w_z \cdot [h_{t-1}, x_t] \tag{2.6}$$

The next step is to calculate the new candidate state or memory of the unit. Equation 2.7 shows how the candidate state is calculated. This is done with the output from the reset gate r_t, which decides how much information from the previous state h_{t-1} to keep.

$$\widetilde{h}_t = tanh(W \cdot [r_t \odot h_{t-1}, x_t] \tag{2.7}$$

The last step is to calculate the new hidden state h_t by performing linear interpolation between the the previous state h_{t-1} and the candidate state $\widetilde{h}_t$. The output of the update

gate z_t decides what information is updated.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{2.8}$$

Due to fewer tensor operations and not having separate memory cells [13], GRU is both more computationally efficient and consumes less memory than LSTM.

2.1.5 Encoder-Decoder Architecture

Encoder-Decoder models have been developed to predict variable-length output sequences. They are specifically designed for sequence-to-sequence prediction problems, where both the input and output are sequences. Encoder-Decoder architecture has seen much traction in Natural Language Processing [14], and has recently been utilized for other prediction problems, such as in air quality forecasting [15], power load forecasting [16], and anomaly detection [17].

The encoder consists of a stack of several recurrent units. Using this type of neural network allows the model to understand the context and temporal dependencies of input sequences. Each unit is responsible for taking a single element of the input sequence, converting it to an internal representation, and propagating it forward. The encoder encapsulates the information from all elements and outputs it in a two-dimensional vector known as the hidden state. The decoder consists of several recurrent units responsible for a prediction y_t at time step t. The initial input to the decoder is the hidden state from the encoder. The output is calculated using the hidden state at the current time step together with respective weights.

One of the significant advantages of encoder-decoder architectures is that input and output length may differ, opening up different use cases such as translation and image captioning.

2.1.6 Attention Mechanisms

Attention has proven useful in natural language processing problems, such as machine translation [18] and abstractive text summarization [19]. An attention mechanism aims to enhance the important parts of input data and fade out the irrelevant parts. Essentially, it assigns a weight to each feature in the input and applies this to each part of the input data according to an activation function. The two most common attention mechanisms are Additive [20] and Dot-Product attention [21]. The main difference between the two relates to how the alignment scores are calculated.

In general, attention mechanisms calculate alignment scores by looking at the interdependence between input and output elements. A variant of attention, called self-attention, calculates the alignment scores only by looking at the input elements. Self-

attention may implement any alignment scoring function such as Additive or Dot-Product attention.

2.1.7 Explainable Artificial Intelligence

The purpose of Explainable AI (XAI) is to make the decisions of AI models understood by humans. XAI can be grouped into three main types:

- Pre-modeling explainability
- Explainable modeling
- Post-modeling explainability

Pre-modeling explainability focuses on explaining the data used to develop models in order to gain a better understanding of the underlying data. Some common methods are exploratory data analysis and explainable feature engineering. Exploratory data analysis can reveal problems such as class imbalance which can then be alleviated. Explainable feature engineering has two main approaches: domain-specific and model-based. Domain-specific feature engineering exploit the knowledge of domain experts in order to select important features. A model based approach to feature engineering takes advantage of various mathematical models to uncover the underlying structure of the dataset, utilizing methods such as clustering.

Explainable modeling aims to use explainable models or utilize joint prediction and explanation. Joint prediction and explanation require that the training dataset includes explanations for each sample. Thus, the model can be trained to make a prediction along with an explanation. Post-modeling explainability has the goal of extracting explanations to describe pre-developed models. Various methods include perturbation mechanisms, backward propagation, and proxy models.

2.2 Literature Review

2.2.1 Time Series Forecasting

In [22], Qing & Niu proposes a scheme for predicting hourly day-ahead solar irridiance values. The prediction scheme uses local meteorological forecasts in order to make solar irridiance forecasts. The proposed model consists of LSTM networks, which consider the dependence of consecutive hours in a day. The model is compared with Linear least square regression and a backpropagation neural network (BPNN). Results show that the proposed model outperforms both competitive algorithms. Compared with BPNN, the proposed LSTM model is 18.34% more accurate in terms of root mean square error (RMSE) when trained with two years of historical data. The LSTM model also shows less

overfitting and a better generalization ability compared with BPNN. Using more historical data for training shows an even larger gap, where LSTM outperforms BPNN by 42.9%.

A recent trend in the field of time series forecasting has been to use models initially developed for Natural Language Processing (NLP) tasks. One such model is the encoder-decoder architecture.

Due et al. [23] proposes an attention-based encoder-decoder model to tackle time series forecasting problems. The authors discuss a problem of traditional encoder-decoder architectures since the encoder must compress the hidden representation into a fixed-length context vector, leading to a gradual decrease in prediction ability as the length of the input time series increases. To mitigate this, the authors propose an attention-based encoder-decoder model. The model is shown to be effective at forecasting multi-step time series values for both long-term and short-term time-step conditions. The proposed model is compared to various baseline models and outperforms methods such as ARIMA, RNN, LSTM, and GRU. The authors attribute the model's success to its ability to learn deep temporal representations and long-term temporal dependency features of multivariate time series data.

Jin et al. [16] propose an attention-based encoder-decoder with GRU in order to predict future electric power load. Power load forecasts are categorized into three types, short-term (daily or weekly), medium-term (one week to one year), and long-term (over one year). The authors focus on the former, arguing that short-term forecasting is the most useful for planning, providing economic management with a reliable decision basis. The network consists of three parts, the encoder with GRU units, the temporal attention layer, and a decoder with GRU units. The proposed network outperforms LSTM, LSTM, and RNN. The authors show that adding attention to an encoder-decoder greatly increases the stability and reduces the variance in error over multiple trials.

Qi et al. [24] proposed a deep neural framework for sales forecasting in E-Commerce. The sales forecasting task is formulated as a sequence-to-sequence learning problem. The authors developed an Encoder-Decoder with self-attention. GRU was chosen for the encoder and decoder because of its higher computational efficiency compared to LSTM. They achieved a wMAPE error as low as 0.455 on a dataset of snack items.

In state-of-the-art research, time series forecasting is mostly done with GRU or LSTM using an encoder-decoder architecture. This approach has been shown to perform well on various time series forecasting problems such as product sales forecasting, power load forecasting, and air quality forecasting.

2.2.2 Interpreting Model Predictions

In [25], Lundberg and Lee propose a unified approach to interpreting model predictions. The proposed method is called Shapley Additive Explanations (SHAP). SHAP is proposed as a feature importance measure that adheres to three properties: local accuracy, missingness, and consistency. Local accuracy means that the sum of feature attributions

should equal the difference between the expected value of the model and the actual prediction, where the expected value is the mean of the model's predictions. The next property states that features missing in the model's input should not impact the SHAP value. The last property states that SHAP values should stay consistent regardless of the ordering of input features. The SHAP values of a model explain the difference between a model's prediction and the expected value of that model. However, the proposed solution requires that the SHAP values are calculated from averaging over all $N!$ possible orderings, which is not feasible for more complex models. The authors describe six methods for approximating SHAP values, where two of them are model-agnostic.

Chapter 3

Methods

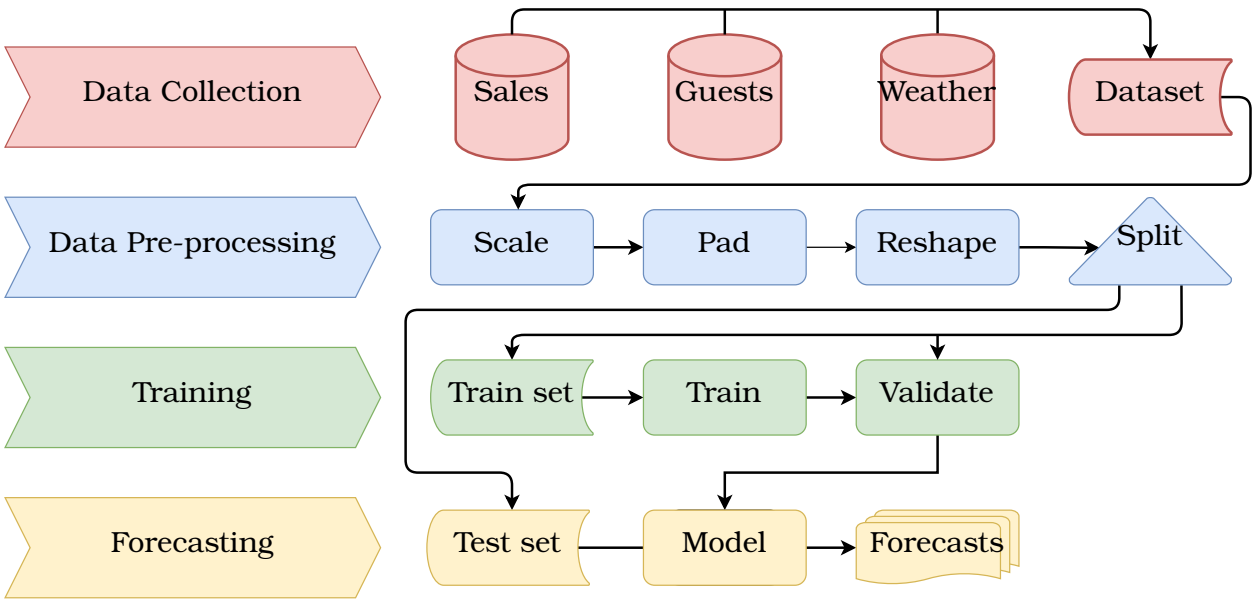

Figure 3.1: Overview of the selected approach.

3.1 Overview

Figure 3.1 shows the general approach. The target variable and predetermined features are retrieved from different databases and APIs during the data collection phase. The next phase is data pre-processing, where the data is scaled, padded, and reshaped before being split up into training and testing sets. The training set is further split into a validation set, which consists of randomly selected samples. The last dataset is the testing set, which is used to evaluate the model's performance. The dataset is further filtered depending on the experiment being carried out. In the next phase, the model is trained on the training set for a given number of epochs, where the training set is run through the model once for each epoch. After each epoch, the performance is evaluated on a validation set to prevent overfitting, where the model optimizes too much on the training data and performs worse on unseen data. Once the model is trained, we evaluate it by predicting the test set, where actual target values are known. This allows predictions made by the model to be compared to the ground truth.

3.2 Dataset

The assembled multivariate time series consists of 12 different features along with the target variable. The id is a unique value assigned to each day of the year by Dyreparken. An id of 42 would correspond to the comparable day in 2018 and 2019, but not the exact same date. This makes it easy to compare days between years since weekdays and holidays can impact turnover. The department id is a unique id assigned to each department by Dyreparken. Datetime is split into year, month, day, hour, and weekday. The public holiday feature is a boolean, where a 1 means that the day is a national holiday or a special event. The daily budgeted guests is a guest budget produced by Dyreparken, one for the zoo and one for the water park. The guest budget usually gives a good approximation of the number of guests that will visit the park, assuming the weather is nice.

- x_0: id
- x_1: department id
- x_1: year
- x_2: month
- x_3: day of the month
- x_4: hour of the day
- x_5: weekday
- x_6: public holiday
- x_7: daily budgeted guests
- x_8: temperature at hour t

- x_9: precipitation at hour t
- x_{10}: lag (sales at hour t previous week)

Temperature and precipitation are the two weather variables selected. Temperature can affect the turnover more locally, and not just due to the seasonal trends. If the temperature is too high, guests may be more likely to go to the water park instead of the zoo. The precipitation accounts for rain and snow. Heavy rains can affect the number of guests that visit the park, and a local rain shower during the day can impact that hour and the consecutive one. The lag is the past turnover one week prior. I.e., the turnover on the same weekday and hour in the previous weeks. Due to the reliance on weather data and the lag value, the model is restricted to forecasting one week into the future.

Chapter 4 analyzes the dataset in more depth.

3.3 Data Pre-Processing

In order to prepare the data for training, it needs to be cleaned. Especially the weather data contains a lot of missing observations. Offline weather stations or sensors could cause this. The temperature data contains a few missing observations. Interpolation has been used to fill the gap by estimating the actual values. Due to more missing observations, the precipitation values have been handled differently. Values have been interpolated one time step in either direction, and the rest are assumed to be 0.

$$z = \frac{x_i - u}{s} \tag{3.1}$$

Deep learning algorithms tend to prefer data that looks like standard distributed data. Therefore, the features are scaled according to a standard distribution as shown in equation 3.1. The scaled value z is calculated by subtracting the mean from the original value x_i and scaling to unit variance s.

$$z = \frac{y_i - y_{min}}{y_{max} - y_{min}} \tag{3.2}$$

The target variable also has to be scaled. We do not want to alter the shape of the data, so a min-max scaling method is used, as shown in equation 3.2. The scaled variable z is calculated by subtracting the minimum value of y y_{min} from the target variable x_i and scaling by the difference. The resulting scaled variable is a value between 0 and 1.

RNNs take only take 3D vector input. Therefore, we must reshape the data into a suitable form. The data is reshaped to (#samples, #timesteps, #features). In case any days have fewer opening hours, the data is padded so that each day contains the same number of time steps.

3.4 Approach

Formally, our task is to learn a regression model which can predict the future temporal value y_i given a set of predetermined features $X_i = [x_1, x_2, ...x_n]$ at hour i. We formulate the problem as structured output prediction in order to predict multiple future time steps. Given an input sequence consisting of hourly feature vectors $X = [X_1, X_2, ..., x_i, \ldots, X_t]$, where X_i is the features at the ith time step, produce an output sequence of hourly turnover values $y = [y_1, y_2, ..., y_i \ldots, y_t]$ where y consist of the hourly turnover for each time step.

We have opted to use an attention-based encoder-decoder architecture to generate sales forecasts. In Chapter 2, we discussed the use of using encoder-decoder architecture for solving time series forecasting problems. Attention-based Encoder-Decoder has performed well on various problems, ranging from short-term power load forecasting to product sales forecasting.

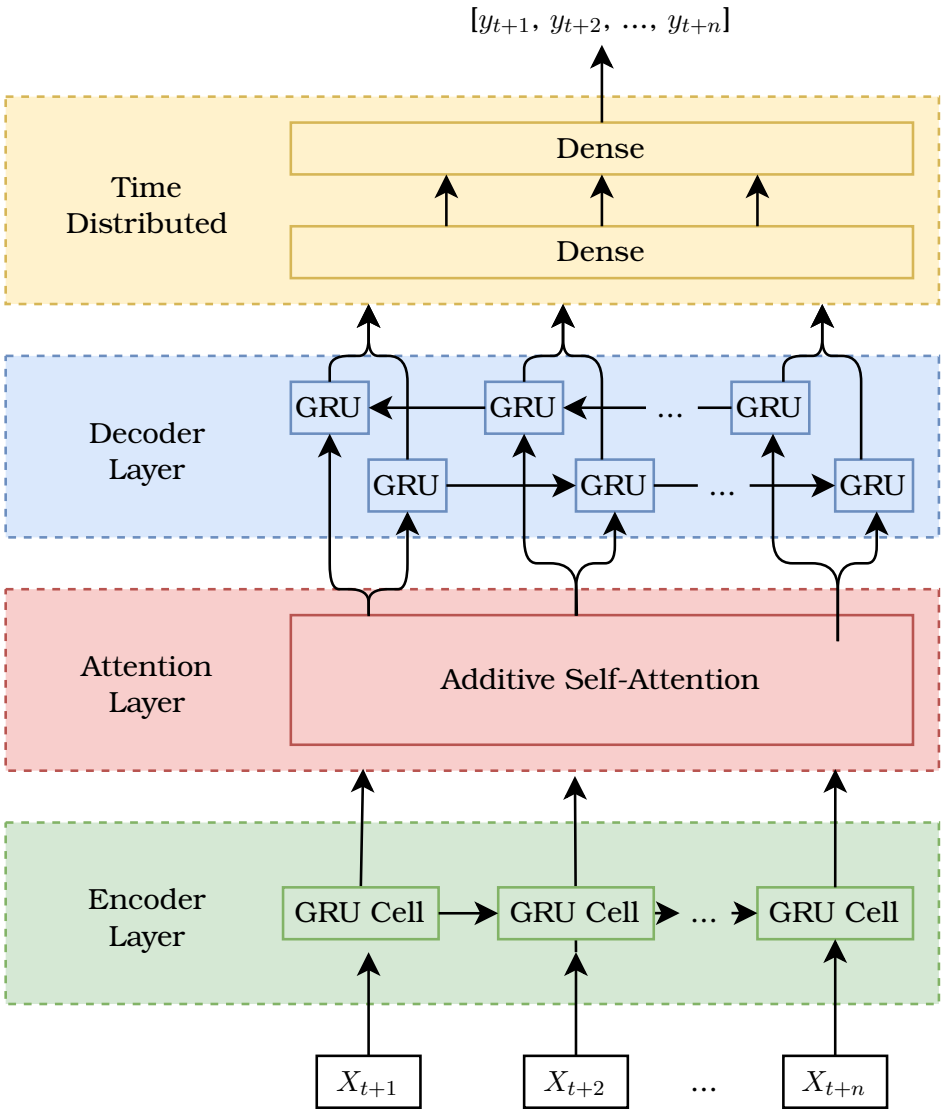

Figure 3.2: Overview of the proposed model.

Figure 3.2 shows the proposed model, which we call GruSeqAtt because it is a type of sequence-to-sequence model with attention and GRU units. The model consists of an encoder, an attention layer, and a decoder. The encoder reads the input sequence, interprets it, and outputs a fixed-length context vector of the model's interpretation. The context vector is repeated once for each time step and passed through the attention layer, which decides what parts of the encoder's output sequence are relevant for the current prediction.

The decoder makes a prediction based on the input received from the attention layer. Finally, the output of the decoder layer is sent to two dense layers with a time-distributed wrapper. The wrapper allows the activation function of the dense layers to be applied individually to each time step.

Attention is provided using a python library [26], which implements self-attention with additive scoring [20]. The attention layer considers the whole context for each time step when performing calculations.

3.5 Shapely-Additive Explanations

We utilize post-modeling explainability to explain the predictions generated by our model. We implement SHAP using the official python library [27]. The library provides a class *DeepExplainer* for explaining the deep-learning model. The class is able to approximate SHAP values by integrating over a set of background samples.

3.6 Performance Metric

Mean Squared Error (MSE) is a commonly used loss function for regression problems. It measures the average of the squares of errors. It helps punish large prediction errors and reduce the influence of small ones. Mean Absolute Error (MAE) is the sum of the absolute differences between the ground truth and the predicted values. Since there is a high variance in sales between days, we opt to use MAE as our performance metric in order not just to prioritize peak sales.

In order to account for differences in the magnitude of sales for each time step, each time step is individually scaled before calculating the mean absolute error. This means that the MAE of a time step is relative to the sales of that time step. The overall error is defined as the average of all time steps t_1 to t_n.

Mean Absolute Percentage Error (MAPE) is a commonly used metric when there is a high variance in the target value. Weighed MAPE (WMAPE) is a version of MAPE modified to overcome the infinite error issue. In our sales forecasting problem, we weigh the error by the total sales volume.

Chapter 4

Data

As described in chapter 3, the problem is formulated as a multivariate forecasting problem, where there are multiple observations for each time step. There are many variables that impact future sales. However, only pre-determined features that are known at prediction time can be used. This chapter outlines the choice of features and performs an investigative analysis of the sales and related features.

4.1 Dataset

Sales data is collected from Dyreparken from 2013 through 2020. The POS reporting database stores past sales data for every location in the park, which will be the target value to predict. The sales data is comprised of each transaction, including a department id, timestamp, and the net amount. Therefore, the sales data is aggregated into hourly sales for each department. Time-related features such as national holidays, weekday, an so on a calculated from the datetime. Special events such as LOS-dagen are retrieved from Dyreparken's own activity calendar.

Data	**Description**	**Type**
Datetime	Time series component	Datetime
isHoliday	Includes national holidays and special events	Boolean
Guests	Budgeted number of daily guests	Int
Temperature	The temperature in Celsius at hour t	Float
Precipitation	Amount of rain in mm at hour t	Float
Net Amount	The turnover at hour t	Float

Table 4.1: The assembled dataset.

The data is summarized in table 4.1. The assembled dataset consists of guest numbers, weather data, and sales data. Hourly sales are retrieved directly from Dyreparken's local database. The guest numbers are retrieved from Dyreparken's budget, which will be used for both training and prediction. The budgeted guest numbers assume that the weather is ideal. However, sometimes that is not the case, and it will negatively impact the actual number of guests. Therefore, weather forecasts can be a valuable metric to explain why sales are low while budgeted guests are high.

The Norwegian weather service YR provides an API to retrieve weather forecasts. However, no database of historical forecasts for Dyreparken is available. Therefore, we will assume that the actual weather is a close enough approximation to the weather forecast and use this for both training and testing. In production, historical data will be used for training, while forecasts are used for prediction. The Norwegian Meteorological Institute provides an API that enables the retrieval of historical weather data from different weather stations. We chose the station at Kjevik because of its proximity to the park and because Kjevik is the station used by YR to display historical weather.

4.2 Data Analysis

Figure 4.1 shows the hourly sales distribution of four different departments. The departments have been chosen to highlight differences between different types of departments. The first is department 2, which is a store located next to the lions. There are multiple outliers at hours 11 and 12. This is caused by the animal presentation of lions at 11:00. This shows how nearby events such as animal presentation affects the sales of a store. The presentation lasts until 11:15, so it appears that guests linger a while after the presentation has ended. It even affects the consecutive hour. Department 64 is a restaurant where it is observed that the turnover almost looks normally distributed. Department 95 is a store located right next to the park entrance. Most of the sales are generated later in the day. This makes sense because guests want to do some last-minute shopping while leaving the park. Department 71 is another store that is affected by animal presentations. However, the time of the presentation changes depending on the time of year. In 2019, the presentations were held at 11:00, 12:15, and 13:00.

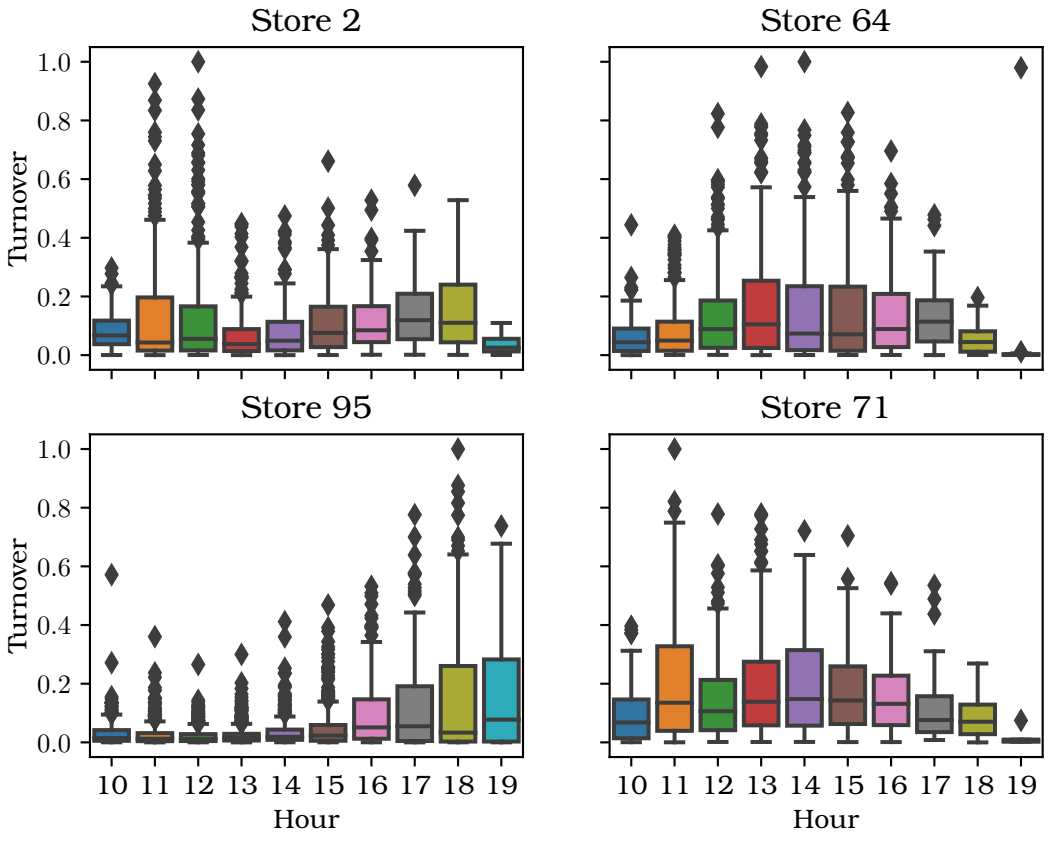

Figure 4.1: Hourly distribution of turnover for the year 2019.

Figure 4.2 shows the monthly distribution of turnover. For a clearer graph, the sales data is aggregated into daily sales and scaled[1]. Department 95 was chosen as the data source since it is open for the entire year. A seasonal trend is clearly visible, where most of the revenue is centered on the summer holiday, with a peak in July. Several outliers are

[1]Due to the confidential nature of sales numbers, all sales data shown are normalized.

visible. There are two distinct outliers in April, both in the Easter break. Similarly, there are two events held in May, which both cause outliers in the distribution. In September, we can see an outlier with a revenue closer to July or August. This is caused by the annual LOS-dagen, where LOS customers gain free entrance.

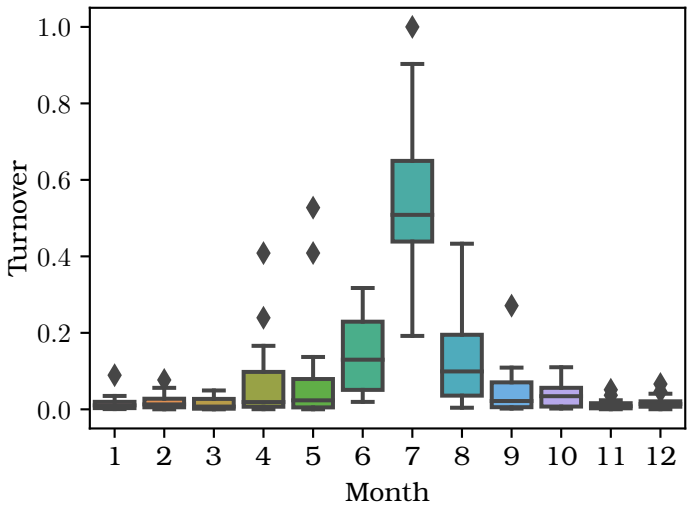

Figure 4.2: The monthly distribution of turnover for the year of 2019.

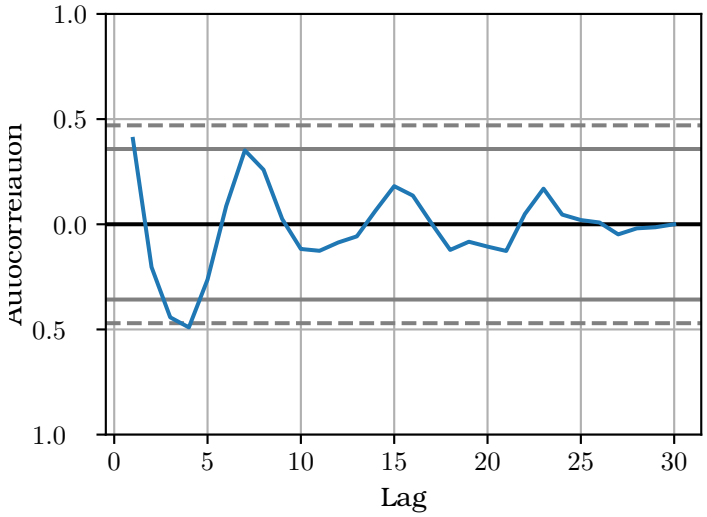

Figure 4.3: Autocorrelation plot of sales and its lag.

Figure 4.3 shows an autocorrelation plot of turnover and its lag. The plot is created from sales data from department 64 at noon in June, but the same low to medium positive correlation can be found with other stores. We observe that there is a medium positive correlation between sales at its seventh lag. This means that there is a significant correlation between sales at a particular hour and the sales from the same hour in the previous week.

Figure 4.4 shows the correlation between features and sales. The number of guests

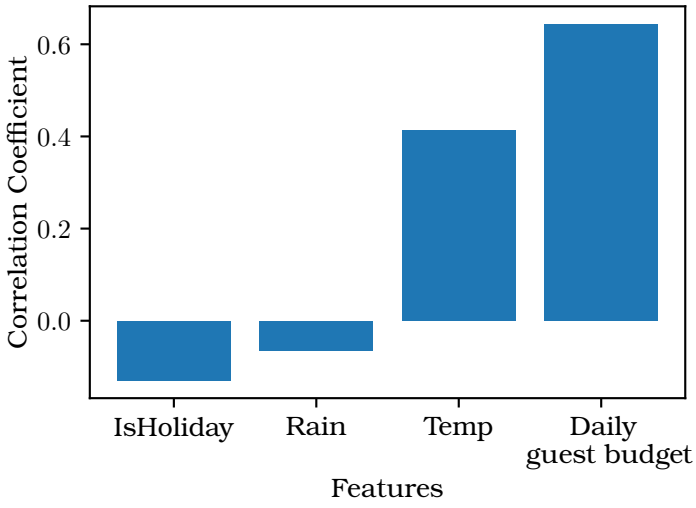

Figure 4.4: Correlation between target value and a set of features.

shows a moderate positive correlation with sales. This is expected as a higher number of guests in the park are more potential customers giving higher sales. Rain shows an insignificant negative correlation with sales, while the temperature shows a significant positive correlation with sales. This is likely explained by the change of seasons, as most of the revenue is generated in the summer months. Interestingly, *IsHoliday* shows a slight negative correlation with sales. This might be due to most of the observations not being on a holiday. The highest-grossing days are in July, which is not covered by *IsHoliday* in order to keep a distinct difference between national holidays, events, and the summer break.

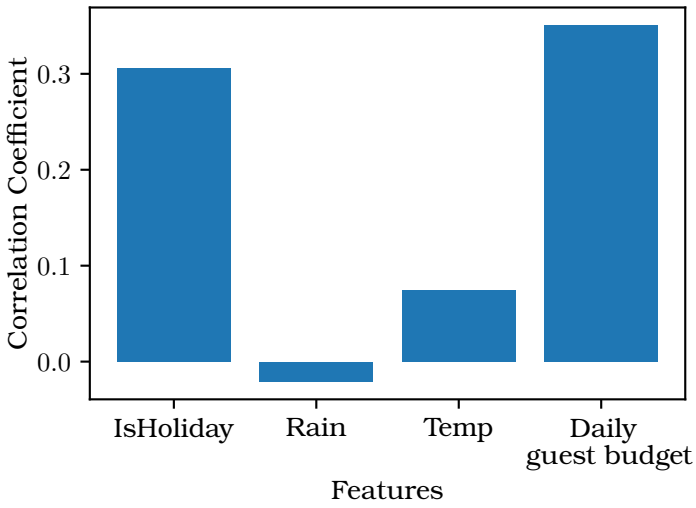

Figure 4.5: Correlation between target value and a set of features using only sales from May.

Figure 4.5 shows the correlation between features and sales in May, which contains a few events and national holidays. We observe a low positive correlation between holidays and

sales. Temp shows a more negligible correlation, likely due to low variation in temperature of the same month. It is also observed that the is a lower correlation with the guests budget for the month of May, caused by lower variance in the number of guests.

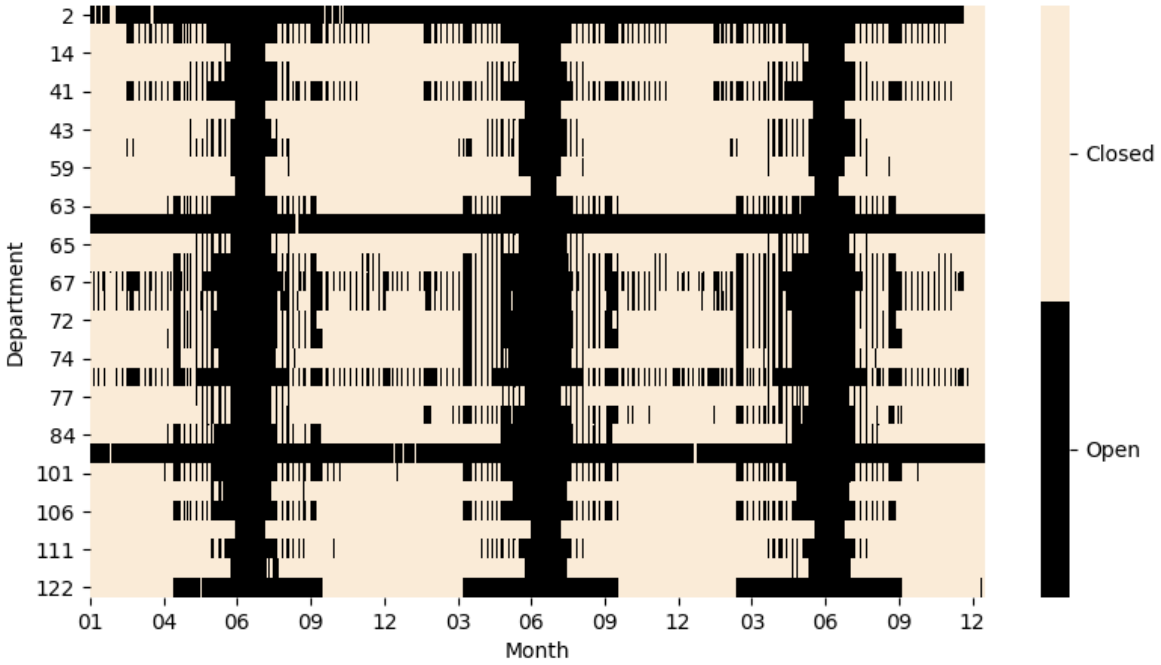

Figure 4.6: Days with sales from 2017 to 2019.

Figure 4.6 shows when locations in Dyreparken are open. It is clear that all departments are open in July. Department 2, 64, and 95 are open all year round. Winter break, Easter break, and Autumn break are visible in February, April, and October, respectively. The third Hypothesis relates to grouping different departments in order to achieve better predictions. This could allow better predictions to be made if a department should decide to open on a day its previously never been open before. Similar departments can be chosen using domain knowledge and trained together. Departments in the same region of the park are affected by the same events and animal presentation. The flow of guests should also be very similar for departments located in the same general area.

Chapter 5

Results and Discussions

This chapter compares the proposed model to a range of baseline models to evaluate the performance. Three different experiments are carried out in order to test our hypothesis. Lastly, we evaluate the proposed model for the 2020 season in order to test how well the model generalizes.

5.1 Experimental Setup

The proposed GruSeqAtt model will be compared to six other baseline models. Table 5.1 shows the hyper-parameters used for the models. Instead of optimizing the parameters of each model, we opted to use the same parameters whenever possible in order to compare the different models fairly.

Model	**Parameter**	**Value**
Baseline	Hidden layers	1
	Units in hidden layers	100
	Dropout	0.2
	Training epochs	100
	Batch size	96
	Loss function	MAE
	Optimizer	Adam
Encoder-decoder	Hidden layers (encoder)	1
	Hidden layers (decoder)	1
	Units in hidden layers	100
	Droput	0.2
	Attention activation function	Softmax
	Training Epochs	100
	Batch size	96
	Loss function	MAE
	Optimizer	Adam

Table 5.1: Hyperparameters of the experimental models.

The XGBoost implementation is provided through the community-developed python library [28]. The number of gradient boosted trees, *n_estimators*, is set to 100.

The Artificial Neural network is a simple shallow neural network with one hidden layer containing 100 units.

The Convolutional Neural Network (CNN) model has a convolutional layer with 64 filters and kernel size 1. It is followed by a max-pooling layer which downsamples the input from the previous layer. The output of the max-pooling layer is then flattened to a 1D vector before being sent into a dense layer.

The LSTM model has a single LSTM with 100 units layer followed by a dense layer with 100 units. The GRU model follows the same pattern.

The LstmSqAtt uses the same implementation as GruSeqAtt, except using LSTM layers for the encoder and decoder.

The hardware and software specifications can be found in appendix A.

5.2 Performance Evaluation

Due to the stochastic nature of the algorithms, we run 30 trials for each model and present the average scores. Occasionally, one of the algorithms will get stuck early on in a local minimum, resulting in a very high error and a flat training error graph. In such cases, the test is rerun. Figure 5.1 shows the results when trained on the department 2 dataset. Data is from 30 trials conducted for each model. XGBoost is not included due to its deterministic behavior. We observe that the CNN model yields the highest error of all. Even vanilla neural networks perform a lot better. GRU and LSTM perform similarly, though GRU has a slightly higher mean. When we use an attention-based encoder-decoder with GRU and LSTM, the results are much more stable, where the figure shows that the variance in the interquartile range is much lower. On the department 2 dataset, the GruSeqAtt model yields the lowest MAE of 0.135 and WMAPE of 0.452.

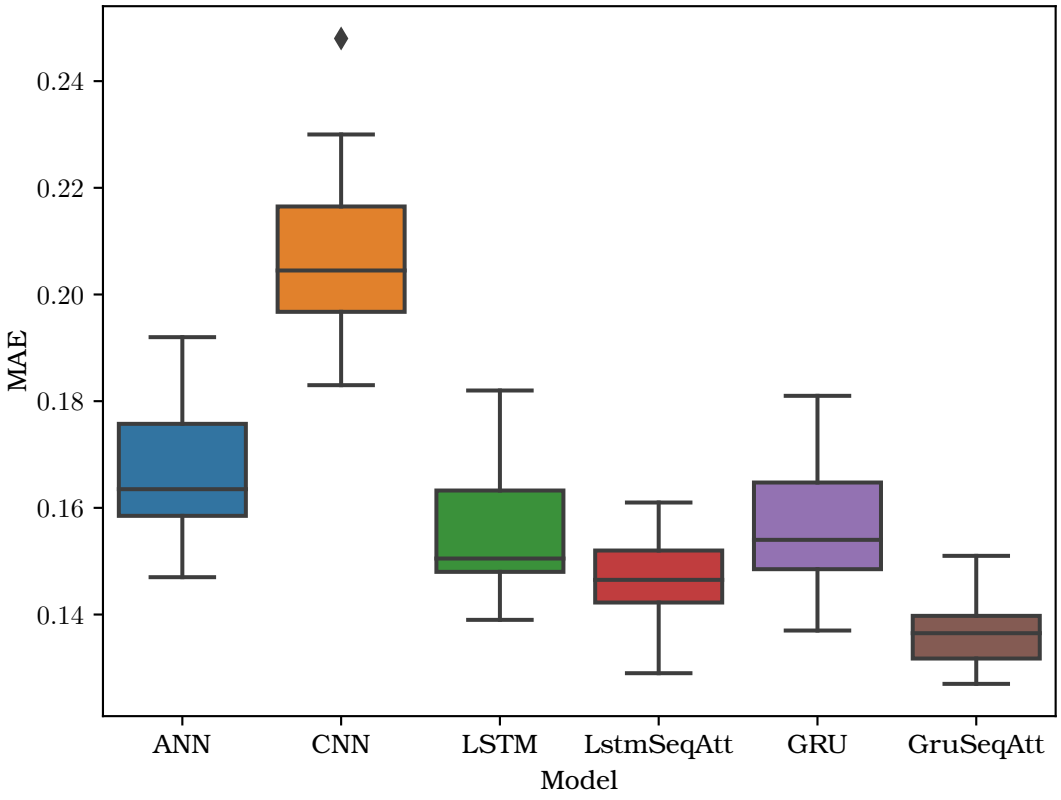

Figure 5.1: Test error (MAE) on store 2 dataset.

Table 5.2 shows the MAE and WMAPE on three different datasets. The first is department 2, a souvenir shop. We observe that GruSeqAtt performs significantly better compared to LstmSeqAtt. The next dataset tested is a restaurant, department 64. Here we observe that LstmSeqAtt and GruSeqATT perform on par. The same is observed for the water park dataset, department 122. Interestingly, GruSeqAtt outperforms LstmSeqAtt on one dataset but performs equally on the other two. To make sure the experiment was carried out multiple times with consistent results.

Model	Retail Store		Restaurant		Water Park	
	MAE	WMAPE	MAE	WMAPE	MAE	WMAPE
XGBoost	0.145	0.484	0.127	0.554	0.123	0.539
ANN	0.170	0.569	0.145	0.572	0.160	0.657
CNN	0.217	0.734	0.170	0.651	0.172	0.704
LSTM	0.155	0.518	0.124	0.508	0.123	0.517
LstmSeqAtt	0.144	0.481	**0.111**	**0.461**	**0.115**	**0.484**
GRU	0.157	0.524	0.124	0.511	0.127	0.532
GruSeqAtt	**0.135**	**0.452**	**0.111**	0.463	0.117	0.493

Table 5.2: Test error (MAE) comparison of Enc-Dec and baseline models. Error is calculated for each time step and average for all time steps. Models are trained with 2013-2018 and tasked with predicting 2019.

Table 5.3 shows the results for each time step. We observe that all models perform poorly on the last time step. A possible cause for this is that department 2 is not open for the full hour, and there is less training data for this time step because it is only open after 19:00 in the summer break. Another cause could be that having a turnover between 19:00 and 19:59 does not necessarily mean that the store has been open at this time. There could have been one sale two minutes after the store was supposed to close. We observe that the model performs the best on time step 11 and 12. This is likely because the most training samples are available for these hours since the store is always open during the animal presentation. The higher error on time step 10 is likely also caused by fewer training samples since the store is only open at this time in July.

Metric	10	11	12	13	14	15	16	17	18	19
XGB	**0.586**	0.325	0.365	0.518	0.428	**0.335**	0.434	0.400	**0.414**	1.039
ANN	0.693	0.395	0.382	0.642	0.523	0.427	0.502	0.487	0.556	1.087
CNN	0.947	0.457	0.522	0.841	0.731	0.680	0.702	0.694	0.758	0.999
LSTM	0.687	0.330	0.381	0.521	0.478	0.373	0.475	0.449	0.534	0.956
LstmSeqAtt	0.632	0.298	0.352	0.521	0.444	0.370	0.431	0.403	0.465	0.894
GRU	0.815	0.332	0.395	0.521	0.468	0.393	0.437	0.430	0.516	0.937
GruSeqAtt	0.587	**0.278**	**0.351**	**0.500**	**0.415**	0.348	**0.415**	**0.382**	0.445	**0.796**

Table 5.3: Test error (WMAPE) on department 2 for each time step.

Overall, our GruSeqAtt model achieves the lowest error compared to the baseline models. If we do not account for the first and last time step, the WMAPE on department 2 is only 0.392. This is comparable to the results made by other researches [24]. Note that results cannot be directly compared due to the use of different datasets.

5.3 Experiment 1: Weather

In [5], Sjursø developed a model to forecast long-term turnover one year in advance. One of the major shortcomings of the model was its inability to predict significant outliers in turnover caused by the weather. Figure 5.2 shows the results from store 41, which was particularly badly hit by rainy weather on the 20th of July. Only 4281 of 9200 budgeted guests visited the park that day. Historical weather data shows that this day had 24mm of rain.

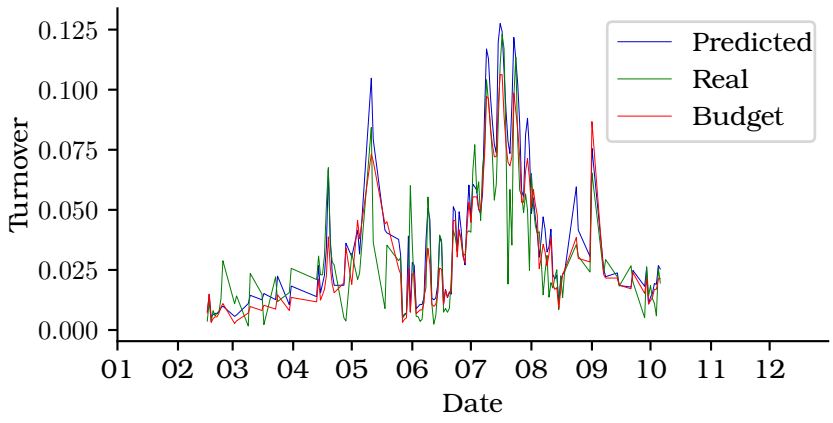

Figure 5.2: Turnover forecast for store 41 [5].

The hypothesis states that adding weather features can help the model create more accurate predictions. In order to test out our hypothesis, we will compare a model trained with and one without weather-related features. Figure 5.3, shows forecasts made in the same year for the summer vacation. Id 729 corresponds to the 20th of July. We observe that on the 20th, the actual sales are much lower than in 2018. Furthermore, in all time steps, the model trained with weather-related features much more accurately predicts the large deviation in turnover caused by rainy weather. However, it often predicts too low turnover. On average, it looks like the model without weather performs better.

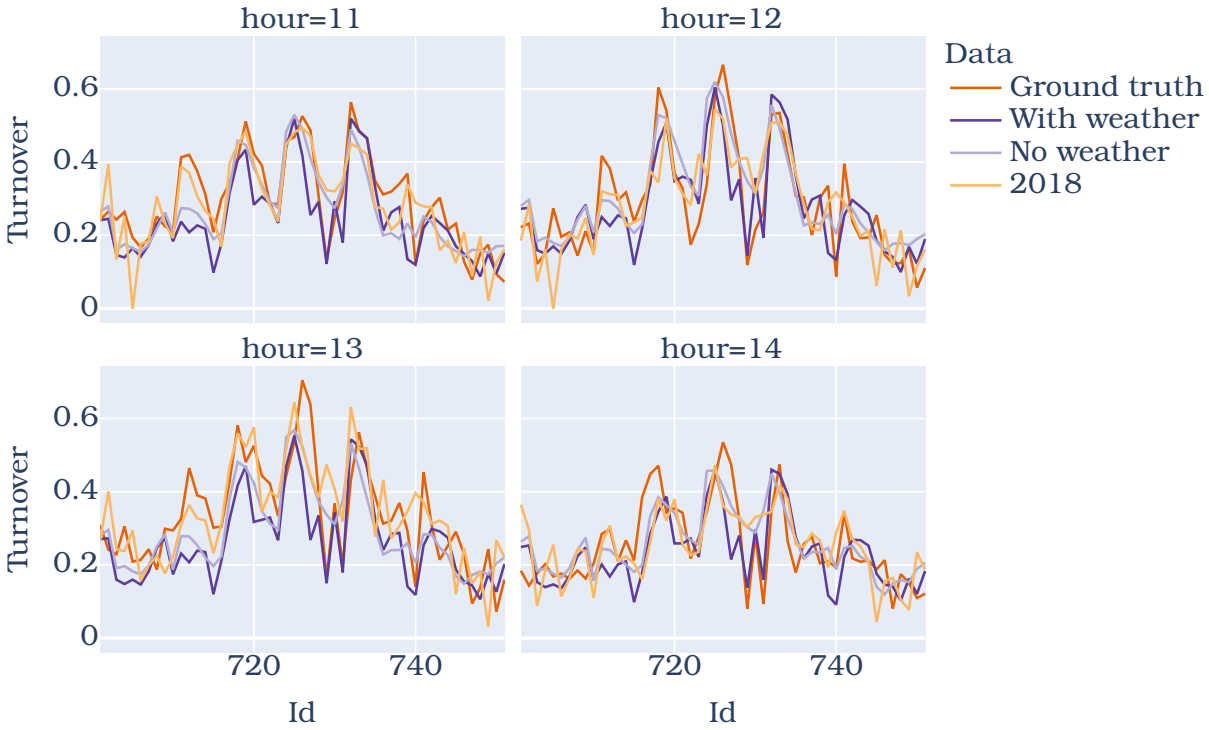

Figure 5.3: Plot of turnover at four different time steps for department 41.

Table 5.4 shows the MAE on the test set for our experiment. We observe that the model

trained without weather-related variables performs better on six out of the seven time steps and significantly better on the first time step. Overall the model with weather features has a slightly higher error. The model with weather features performs the same or worse on all except the last time step.

Features	**Avg**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
No weather	0.158	0.116	0.106	0.127	0.133	0.160	0.155	0.307
With weather	0.161	0.142	0.114	0.134	0.139	0.161	0.156	0.280

Table 5.4: Test error (MAE) comparison from weather experiment.

It appears that the rain variable can act as noise to the model. The effects of rain on turnover depend on several things. Too much rain might result in fewer people visiting the park, in turn causing a lower turnover. However, if there is just rain for one hour, the turnover might increase for that hour since guests want to go inside. This could also affect the following hours, where guests want to spend time outside since the weather has cleared up. To make things more complicated, departments will behave differently to the rain variable, depending on if it is a store, a takeaway, or a sit-in restaurant.

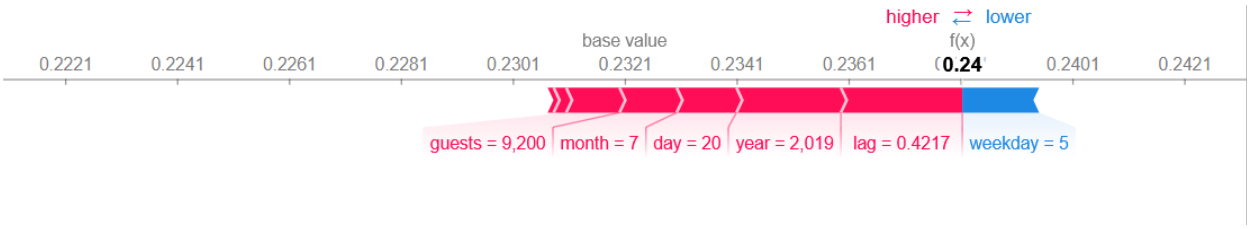

Figure 5.4: SHAP force plot of prediction made by the model without weather features.

We will utilize SHAP to explain the reasoning behind the predictions made by the two different models. Figure 5.4 shows a force plot of the prediction made of July 20th at 13:00 using the model without weather-related features. The base value is the mean of predictions, i.e., the value that would have been predicted without the impact made by features. The features in red push the value higher than the base value, while the features in blue push the base value lower. The width of the feature represents the impact on the prediction. A wider feature has a more significant impact. We observe that the most important feature is the number of budgeted guests, increasing the predicted value compared to the base value. Weekday has a negative impact on the predicted value, which is logical since Saturdays are not the highest-grossing weekday in the summer break.

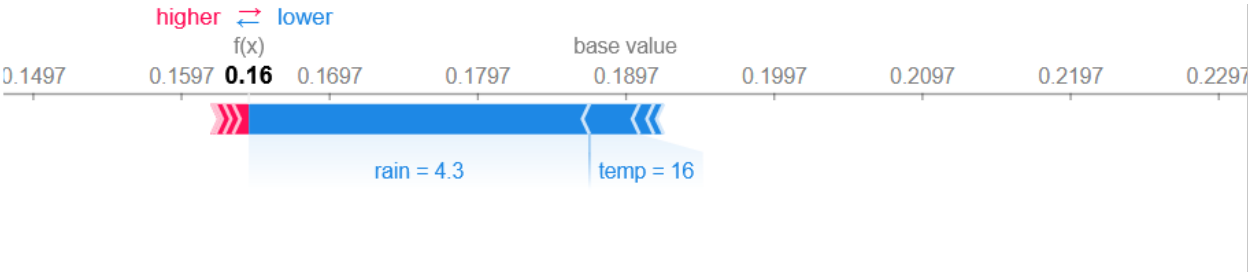

Figure 5.5: SHAP force plot of prediction made by the model with weather features.

Figure 5.5 a *force plot* of the prediction made by the model with weather features. We observe that the most important feature is rain, which significantly lowers the predicted value from the base value. The second most important feature is the temperature, which is lower than usual because of the cloudy weather. It is also observed that the model with weather features has a lower base value. This coincides with what was observed in figure 5.3.

Overall, the model appears to fail at predicting some of the higher peaks. However, it can more accurately predict large deviations caused by weather effects. Predicting large drops in turnover enables Dyreparken to make appropriate changes. Such as deploying staff to other locations which is not affected as much or cancel shifts. This is very useful for amusement parks and zoos to prevent overstaffing. The use of SHAP to explain predictions made by a model can be beneficial to gain insight into the choices made by a model. Moreover, it can help users gain more trust in predictions. Although our data analysis showed that rain has a low correlation with sales in general, it can be critical to consider for a specific prediction.

5.4 Experiment 2: Summer Break

The second hypothesis states that it is possible to lower prediction error on summer break forecasts if the model is only trained with data on the summer break. The hypothesis will be tested by comparing a model trained with only data on previous summer breaks and a model trained with data for the entire year. To conduct our experiment, we will choose the dataset on department 2, which is open most of the year. Time steps are reduced to 10:00-18:00 in order to reflect the stores opening hours.

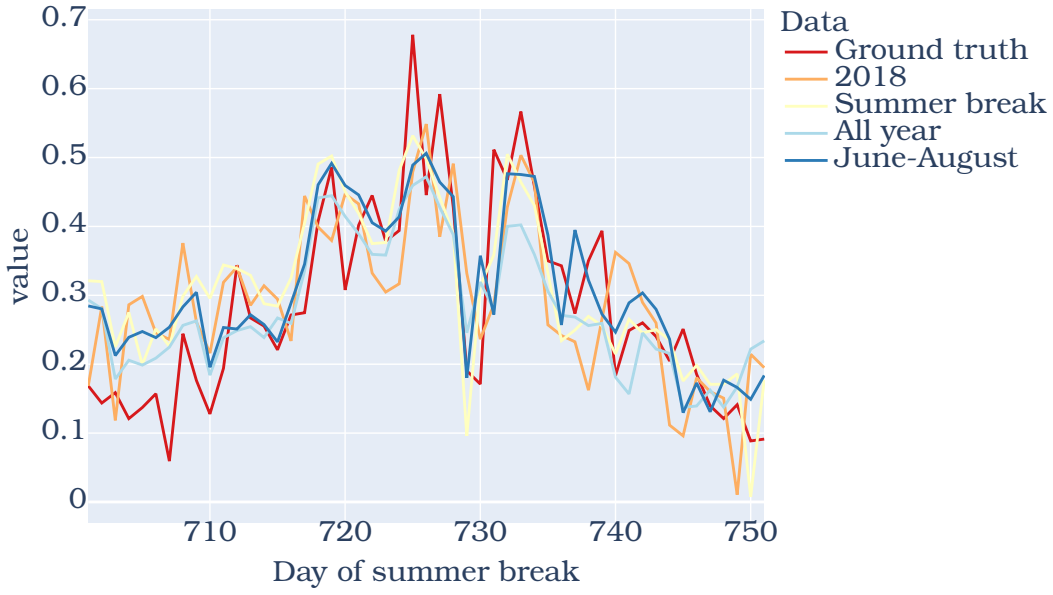

Figure 5.6: Plot of actual turnover and forecasted turnover at 13:00.

Figure 5.6 shows the three experiments plotted along with the ground truth and turnover for 2018. In 2019, the Summer break was from the 22nd of June to the 11th of August. The plot shows days ordered from the 1st day of summer until the last. This way, we can compare 2019 with 2018. It is observed that all three provide similar predictions. They all predict a bit high turnover at the start of the summer break, somewhere in between 2018 and 2019 turnover. None of them accurately predict the highest peaks in the middle of July, though the turnover is slightly higher than in 2018. They all predict the sharp decrease in turnover around id 729, with the model trained from June to August providing the best fit.

Training Data	**MAE**	**WMAPE**
Summer Break	0.144	0.399
June-August	0.138	0.380
All year	0.138	0.380

Table 5.5: Test error comparison from the summer break experiment.

Table 5.5 shows the results of the experiment. We observe that including training data from June and August, in addition to the summer break, decreases the prediction error. No significant change is observed from adding more training data beyond June and August.

Although our results show that training a model only to generate summer forecasts does not improve predictions, it does show that we can improve them by using more data. Our model can still learn something useful from the training data, even if that data is not from the same period as predictions are being made. Also, having more training data from other periods of the year does not negatively impact summer forecasts.

5.5 Experiment 3: Cold Start

Hypothesis 3 states that a model can achieve better predictions where no training data is available if trained together with another store. This can be useful if a department has limited training data, is increasing its opening hours, or is going to open on a public holiday it has never been open before. To test out this hypothesis, we choose departments 75 and 101. They are both large souvenir shops located in different regions of the park. They are both open in the Easter and Autumn break, which allows us to compare our results with actual turnover.

The first test investigates whether the model can make better Easter break predictions on store 75 if trained together with store 101. To prepare our dataset, we combine the data from both departments but remove all training data in the Easter from store 75. This experiment aims to predict sales for department 75 without having any training data for the Easter break. For the first experiment, three different training sets are tested:

- Only department 75 (no Easter data)
- Department 75 and 101 (no Easter data)

- Department 75 (no Easter data) and 101 (with Easter data)

Table 5.6 shows the results from the first experiment. The error is calculated from the test set, which only contains the Easter break for department 75. The first case achieves an MAE of 0.251. Adding department 101 without Easter decreases the error by 0.026. With department 101, including Easter, the error is decreased by a total 0.110. We observe that most of the reduction in error can be attributed to adding training data on the Easter break.

Data (Easter Y/N)	**Avg**	**11**	**12**	**13**	**14**	**15**	**16**
75(N)	0.251	0.325	0.213	0.257	0.260	0.231	0.219
75(N)+101(N)	0.225	0.300	0.193	0.231	0.231	0.200	0.193
75(N)+101(Y)	0.141	0.208	0.108	0.134	0.142	0.118	0.135

Table 5.6: Test error (MAE) on the Easter break.

Table 5.7 shows the results for the Autumn break. The same trend is observed, where adding training data from another store significantly decreases the prediction error. Although the improvement is not as large as with the Easter break, it is still significant.

Data (Autumn Y/N)	**Avg**	**11**	**12**	**13**	**14**	**15**	**16**
75(N)	0.259	0.333	0.168	0.201	0.168	0.302	0.383
75(N)+101(Y)	0.227	0.306	0.157	0.194	0.161	0.258	0.282

Table 5.7: Test error (MAE) on the Autumn break.

These results show that using data from other departments can decrease prediction errors. However, due to significant differences between departments, they should be chosen carefully using domain knowledge. In particular, a takeaway restaurant can respond very differently to precipitation compared to a sit-in restaurant.

5.6 Generalizability

To test the generalizability of the model, we will predict 2020. Because of pandemic restrictions, a lot changed during the 2020 season. These factors could make it difficult for a model trained on pre-pandemic data. The following factors can influence 2020 forecasts:

- Maximum limit on the number of guests in the park at any given time
- To avoid large queues, each guest was assigned a slot time for when they could enter the park.
- A new ticket option was made available with entrance in the afternoon
- A limit to the number of guests present at animal presentations.
- Some restaurants were changed from sit-in to take-away.

- Due to restrictions, restaurants had to stop serving buffets.
- Restaurants only allowed food to be ordered through the app. Possibly affecting the behavior of customers.

Unfortunately, when the lockdown started, no adjusted guest budget was made for 2020. The only guest budget available is the one made at the end of 2019. Therefore, we will test both with the original budget and a budget where the number of guests is capped at 11000 guests. The data is split into a training set ranging from 2013 to 2019 and a test set with data from 2020.

	Avg	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Original budget	0.158	0.174	0.177	0.105	0.104	0.114	0.182	0.208	0.202
Adjusted budget	0.159	0.185	0.188	0.109	0.105	0.113	0.173	0.201	0.199

Table 5.8: Test error (MAE) on 2020 test set for department 2.

Table 5.8 shows the MAE on the 2020 test set. We observe that the average MAE is 0.023 higher compared to the experiment conducted in 2019. Predictions using the adjusted budgets appear to perform worse in the earlier hours and better in the later hours of the day. The adjusted budget does not perform better because there is only a short period in the summer that the guest budget exceeds 11000. Another reason could be that capping the budget flattens the distribution of the data. A better approach could be to scale the guest budget down.

	Avg	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Original budget	0.605	0.770	0.749	0.599	0.560	0.534	0.539	0.575	0.517
Adjusted budget	0.612	0.817	0.793	0.620	0.565	0.528	0.511	0.556	0.509

Table 5.9: Test error (WMAPE) on 2020 test set for department 2.

Table 5.9 shows the same results in the form of WMAPE. This shows that the MAE metric can be a bit misleading. Looking at the MAE table, the earlier hours appear to perform better than later hours of the day. The WMAPE scores show that the model performs the worst on the first two time steps. The reason is that the lion presentation happens at 11 O'clock. As shown in the chapter 4, the animal presentation significantly affects the turnover for both time step 11 and 12. Due to the pandemic restrictions, the number of guests attending animal presentations is capped at 200, which leads to the model significantly overestimating the turnover at these hours. There is currently no way to adjust for this in the model.

Metric	**Avg**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
MAE	0.156	0.286	0.110	0.125	0.116	0.101	0.164	0.151	0.192
WMAPE	0.551	1.075	0.452	0.384	0.404	0.415	0.448	0.527	0.706

Table 5.10: Test error on 2020 test set for department 64.

Table 5.10 shows results from store 64. The model achieves an average error of 0.156 in terms of MAE, 0.045 higher than the prediction in 2019. The model struggles at predicting the first time step, possibly due to the high variance of turnover at this hour.

5.7 Summary

This chapter has focused on evaluating the proposed model's performance and conducting experiments to test our hypotheses. We conducted three different experiments related to the hypotheses and an experiment to evaluate how well the model generalizes. We compared the proposed model, GruSeqAtt, with a range of different models. Results show that the GruSeqAtt model outperforms all other models on the retail store dataset. With the restaurant and water park datasets, GruSeqAtt shows comparable accuracy to the LstmSeqAtt model. The GruSeqAtt model achieves the lowest error on the retail store dataset on seven out of ten time steps.

To see how well the model generalizes, it was tasked to forecasts the 2020 season for department 2. Results show that the average MAE is 0.023 higher compared to the 2019 forecasts. For department 64, the average MAE was 0.045 higher compared to the 2019 forecasts. More work must be done to address the changes made due to the pandemic to create more accurate forecasts.

Hypothesis 1: *Using past weather forecasts can increase the accuracy of sales forecasts in a Zoo.*

Hypothesis 1 claims that including weather in the forecasting model will increase the accuracy of sales forecasts in a Zoo. The reasoning behind the hypothesis is that rain can negatively impact the number of visitors in the park.

Our experiment has shown that adding weather features to the model increases the overall prediction error. However, using weather features enables large deviations in turnover caused by poor weather conditions to be detected. A possible cause for this was discussed. Weather likely acts as noise to the model. If it rains an entire day, that will likely lower the number of guests visiting the park. However, if there is just rain for one time step, it might just affect the turnover at the current and consecutive time step, depending on the department type.

In this sector, there is a high risk associated with the weather. To mitigate this, businesses have the ability to send employees home while compensating them for their lost time. Being able to detect instances where the weather might impact workload allows the business to take action beforehand. Shifts can be canceled ahead of time, saving employees the trouble of commuting to work just to be sent home.

Hypothesis 2: *A model trained to only forecast the Summer break can yield more accurate predictions on summer forecasts than a model trained to forecast the entire year.*

The second hypothesis claims that a model can produce more accurate forecasts if the model is only trained on the part of the year it is supposed to predict. The goal of the experiment was to predict the summer break while training the model on three different sets:

- A model trained on the entire year

- A model trained on June to August
- A model trained on only the summer break

Contrary to the hypothesis, the results showed that the model trained with data from June to August achieved significantly lower error than the model trained on only the summer break. Using the entire year as training data did not significantly impact the results, showing that adding more training data not directly related to the period to predict does not negatively impact the results.

Hypothesis 3: *It is possible to successfully pre-train a model with data from one location to increase the accuracy of revenue predictions of another.*

The last hypothesis states that a model can be trained with data from other departments to increase the accuracy of forecasts for a specific department. This can be especially useful if a department opens in a period where there is no prior training data. The experiment was carried out two using training data from two different souvenir shops located in different park regions. The experiment's goal was to predict the Easter and Autumn break for department 75 without any prior training data on the holidays for the given department.

Results show that adding training data from another department significantly decreases the forecasting error. Most of the gain comes from including training data for the same holiday from the other department. The training data from other periods also decreases error, strengthening the findings from the experiment related to hypothesis 2.

Chapter 6

Conclusions and Further Work

This chapter concludes the thesis and discusses possible future work to improve sales forecasts.

6.1 Conclusions

This thesis aimed to explore the use of deep-learning models for hourly sales forecasts in zoos and amusement parks. Sales forecasting is a complex problem due to the high variance of sales data. In addition, the domain provides some unique aspects which must be considered due to the multiple sites, differing opening hours, and dependence on the number of visitors. Two goals were established. Firstly, to assemble a multivariate dataset suitable for sales forecasting, and secondly, to develop a model capable of generating hourly sales forecasts. We tackled the problem as a structured output prediction problem, where the goal of the model was to predict a sequence of turnover for each time step of the day, given a sequence of feature vectors. The proposed model consists of an attention-based encoder-decoder with gated recurrent units.

Results show that the proposed model outperforms five of the baseline models while achieving comparable or better results than the attention-based encoder-decoder with LSTM. The proposed model also provided the most stable results, with the lowest variance between trials. When evaluated on the 2020 pandemic dataset, the model performed between 15% and 29% worse than the 2019 forecast. More work must be done to create a model that can create accurate sales forecasts despite pandemic measures.

Results related to the first hypothesis show that adding weather features results in a slightly higher prediction error. While this decreases the accuracy of forecasts, it enables large deviations caused by weather effects to be detected. This is useful for businesses in the domain of amusement parks and zoos since it mitigates risk associated with weather. SHAP proved to be a useful tool for explaining the reasoning behind model predictions. Results from the second hypothesis indicate that a model only trained to forecast the summer does not yield better predictions than a model trained to forecasts the entire year. However, it shows that models perform better with more training data, even if that data is not from the same period as the forecast. This is useful because it enables data from lower activity periods to boost prediction performance in the high activity periods such as the summer break. The last experiment shows that models can be improved using more training data from similarly behaving departments selected using domain knowledge. This will allow more accurate forecasts to be generated and be helpful if training data is lacking. Such as if a department is expanding its opening hours and has no prior training data for that time period.

6.2 Further Work

6.2.1 Pandemic Measures

When applying our model to the 2020 dataset, we discovered that changes done as a response to the pandemic have significantly impacted the accuracy of predictions. Several measures should be investigated to improve the model's generalizability and allow 2020 and 2021 data to be used for training. However, any changes to the model should not negatively impact forecasts for regular seasons. A significant factor in 2020 was the introduction of slot times, where guests can only enter at their allocated slot times. This has worked so well at alleviating queues that Dyreparken is considering making this change permanent. A possible solution to this is to use hourly budgeted guests instead of daily guests. Historical budgets can still be used by making an hourly distribution according to the distribution of actual guests in the park. Future budgets can be made by distributing the guest budget according to slot times. Making an hourly distribution of budgeted guests can allow more flexibility to deal with pandemic measures.

6.2.2 Model Flexibility

In amusement parks and zoos such as Dyreparken, major changes happen every season. New departments open up, others expand or change. In order to deal with this, the model should be made more flexible. A possible solution is to include information about the department in the model, such as the department's location and type. This could allow the model to better learn the differences between departments and how they behave to different features. This could allow more departments to train together, thus allowing for more accurate predictions to be made if changes are made to a department, or a new one opens up. Another benefit of this would be that training data would still be valid even if significant changes are made to a department.

As shown in chapter 4, animal presentations and other park activities can significantly impact turnover. The results on the department 2 dataset have shown that the model can accurately predict the higher turnover when animal presentation occurs. However, that would no longer be the case if the time of the animal presentation changed. By including a simple Boolean value for whether or not an animal is taking place in the current hour, we could mitigate this problem.

6.2.3 Weather

The experiment related to our first hypothesis has shown that the rain feature can confuse the model due to the complexity of how rain relates to sales. Results showed that adding weather-related features enables large drops in turnover to be detected. However, it increases the overall error. A possible improvement could be to include both hourly and daily aggregated precipitation to better distinguish between rain that causes fewer

visitors or rain that have a more local effect.

6.2.4 Hyper-Parameter Tuning

In order to fairly compare the different methods, none of them were exhaustively tuned. Only simple trial and error parameter tuning was performed. There could be more performance to be gained by performing hyper-parameter tuning on the model. A possible approach is to use Bayesian optimization in order to tune the model.

Appendix A

Hardware and Software Environment

Operating System	Windows 10 20H2 19042.685
Processor	AMD 5900X
Memory	32GB DDR4
Graphics	1x NVIDIA GeForce RTX 2080 8GB

The following versions where used to conduct the experiments.

- CUDA 10.1
- Python 3.7
- Keras v2.4.3
- Tensorflow v2.4.1
- XGBoost v1.4.1

Bibliography

[1] Dyreparken. *Besøkstall*. URL: https://www.dyreparken.no/om-dyreparken/besokstall/.

[2] Manish Shukla and Sanjay Jharkharia. "ARIMA models to forecast demand in fresh supply chains." In: *International Journal of Operational Research* 11 (2011), pp. 1–18.

[3] Qian Wen et al. "Daily Sales Forecasting for Grapes by Support Vector Machine." In: *Computer and Computing Technologies in Agriculture VII*. Ed. by Daoliang Li and Yingyi Chen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 351–360. ISBN: 978-3-642-54341-8.

[4] F.M. Thiesing and O. Vornberger. "Sales forecasting using neural networks." In: *Proceedings of International Conference on Neural Networks (ICNN'97)*. Vol. 4. 1997, 2125–2128 vol.4. DOI: 10.1109/ICNN.1997.614234.

[5] Olav Markus Sjursø. *A Neural Network Approach to Turnover Prediction for a Norwegian Zoo. Report in IKT442, University of Agder.* 2020.

[6] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[8] Mike Schuster and Kuldip Paliwal. "Bidirectional recurrent neural networks." In: *Signal Processing, IEEE Transactions on* 45 (Dec. 1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[9] Victor Carbune et al. "Fast Multi-language LSTM-based Online Handwriting Recognition." In: *International Journal on Document Analysis and Recognition (IJDAR)* (2020).

[10] Iram Shahzadi et al. "CNN-LSTM: Cascaded Framework For Brain Tumour Classification." In: *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*. 2018, pp. 633–637. DOI: 10.1109/IECBES.2018.8626704.

[11] Weicong Kong et al. "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network." In: *IEEE Transactions on Smart Grid* 10.1 (2019), pp. 841–851. DOI: 10.1109/TSG.2017.2753802.

[12] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.” In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://www.aclweb.org/anthology/D14-1179.

[13] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling.” English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[14] Kyunghyun Cho et al. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches.” In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://www.aclweb.org/anthology/W14-4012.

[15] Leiming Yan et al. “Encoder-Decoder Model for Forecast of PM2.5 Concentration per Hour.” In: *2018 1st International Cognitive Cities Conference (IC3)*. 2018, pp. 45–50. DOI: 10.1109/IC3.2018.00020.

[16] Xue-Bo Jin et al. “Deep-Learning Forecasting Method for Electric Power Load via Attention-Based Encoder-Decoder with Bayesian Optimization.” In: *Energies* 14.6 (2021). ISSN: 1996-1073. DOI: 10.3390/en14061596. URL: https://www.mdpi.com/1996-1073/14/6/1596.

[17] Edan Habler and Asaf Shabtai. “Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages.” In: *Computers Security* 78 (2018), pp. 155–173. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2018.07.004. URL: https://www.sciencedirect.com/science/article/pii/S0167404818303729.

[18] Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation.” In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

[19] Alexander M. Rush, Sumit Chopra, and Jason Weston. “A Neural Attention Model for Abstractive Sentence Summarization.” In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 379–389. DOI: 10.18653/v1/D15-1044. URL: https://www.aclweb.org/anthology/D15-1044.

[20] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate.” In: 3rd International Conference on Learning Representations, ICLR. Jan. 2015.

[21] Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation.” In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

[22] Xiangyun Qing and Yugang Niu. "Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM." In: *Energy* 148 (2018), pp. 461–468. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2018.01.177. URL: https://www.sciencedirect.com/science/article/pii/S0360544218302056.

[23] Shengdong Du et al. "Multivariate time series forecasting via attention-based encoder–decoder framework." In: *Neurocomputing* 388 (2020), pp. 269–279. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.12.118. URL: https://www.sciencedirect.com/science/article/pii/S0925231220300606.

[24] Yan Qi et al. "A Deep Neural Framework for Sales Forecasting in E-Commerce." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. Beijing, China: Association for Computing Machinery, 2019, pp. 299–308. ISBN: 9781450369763. DOI: 10.1145/3357384.3357883. URL: https://doi.org/10.1145/3357384.3357883.

[25] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[26] CyberZHG. *Keras Self-Attention*. https://github.com/CyberZHG/keras-self-attention.

[27] Scott Lundberg. *SHAP (SHapley Additive exPlanations)*. https://github.com/slundberg/shap.

[28] Distributed (Deep) Machine Learning Community. *XGBoost eXtreme Gradient Boosting*. https://github.com/dmlc/xgboost.