HALMSTAD UNIVERSITY

Degree project Computer Science and Engineering, 300 credits

MASTER THESIS

Forecasting EV Charging Station Occupancy Using ST-GCN And Adapter-Based Transfer Learning

Improving charging station occupancy forecasting on ChargeFinder data using spatial-temporal graph convolutional networks and introducing BAM, an adapter for efficient fine-tuning

Daniel Persson & William Wahlberg

Halmstad University, December 16, 2024–version 1.0

Daniel Persson & William Wahlberg: *Forecasting EV Charging Station Occupancy Using ST-GCN And Adapter-Based Transfer Learning,* Improving charging station occupancy forecasting on ChargeFinder data using spatial-temporal graph convolutional networks and introducing BAM, an adapter for efficient fine-tuning, © August 2024

ABSTRACT

As electric vehicle (EV) adoption increases, the demand for efficient and reliable charging infrastructure becomes increasingly important. Accurately forecasting charging station occupancy is useful for optimizing the use of existing infrastructure and improving the overall user experience. This study utilizes data from ChargeFinder, focusing on Swedish charging stations, and applies various data processing techniques to manage missing and intermittent data, as well as cleaning the data to produce a robust input for the model.

To predict occupancy, this thesis implements a Spatio-Temporal Graph Convolutional Network (ST-GCN) that incorporates a graph convolutional network (GCN) as the spatial component and a Transformer model as the time series component. Additionally, the thesis introduces the Bridged Attention Module (BAM), an adapter-based transfer learning method designed to facilitate parameter-efficient fine-tuning across different city networks, even with limited data.

Experimental results, conducted on data provided by ChargeFinder, revealed that the ST-GCN with the incorporated Transformer component produced the best overall performance, highlighting the strength of this architecture for predicting EV charging station occupancy. Furthermore, the results demonstrate the effectiveness of the BAM adapter, which not only achieved competitive Mean Squared Error (MSE) performance but also outperformed baseline models in several scenarios while using a small number of trainable parameters. These findings underscore the potential of integrating spatial and temporal components into the forecast model with adaptive transfer learning techniques to enhance the scalability and reliability of EV charging infrastructure management.

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to our supervisors, *Anna Vettoruzzo* and *Slawomir Nowaczyk*, for their invaluable guidance and support throughout this research. Their insightful feedback and comments were instrumental in shaping this work. We also extend our sincere thanks to *Jonas Blanck*, co-founder and CEO of ChargeFinder, for his availability and for providing the data used in this research project. Additionally, we would like to thank *Ali Amirahmadi* for his feedback regarding the Transformer model in this project. Lastly, we would like to thank the rock band *Rammstein* for producing music that has been listened to many hours over the course of this project.

Best regards,
Daniel Persson & William Wahlberg

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AR	Auto Regressive
ARIMA	Auto Regressive Integrated Moving Average
BAM	Bridged Attention Module
CNN	Convolutional Neural Network
DL	Deep Learning
EV	Electric Vehicle
GCN	Graph Convolutional Network
GNN	Graph Neural Networks
GRU	Gated Recurrent Unit
k-NN	k-Nearest Neighbor
LLM	Large Language Model
LOCF	Last Observation Carried Forward
LSTM	Long Short-Term Memory
LVCF	Last Value Carried Forward
MICE	Multivariate Imputation by Chained Equations
MA	Moving Average
MSE	Mean Squared Error
MLP	Multi-Layer Perceptron
NOCB	Next Observation Carried Backwards
RNN	Recurrent Neural Network
SMHI	Swedish Meteorological and Hydrological Institute
ST-GCN	Spatio-Temporal Graph Neural Network
TPE	Tree-structured Parzen Estimator
Scenario 1	Värnamo-Varberg
Scenario 2	Varberg-Malmö

1 INTRODUCTION

The electric vehicle (EV) industry is witnessing rapid global growth. In 2022, EV sales surpassed 10 million, with their share of total vehicle sales rising from around 4% in 2020 to 12% in 2022 [1]. This surge marks a significant transition in the transportation sector, moving from fossil-fueled vehicles to electric-powered alternatives.

In Europe, this momentum is driven by the European Green Deal, which aims to steer the continent toward climate neutrality and achieve net-zero greenhouse gas emissions by 2050 [2]. This goal is further reinforced by an agreement between the European Parliament and Council, mandating that all new cars and vans in Europe be emission-free by 2035. As part of this transition, updated CO_2 standards require a 55% reduction in emissions for new vehicles and a 50% decrease for new vans by 2030 [3]. In line with these targets, companies such as Volvo, Volkswagen, and KIA have pledged to transition to fully electric models by 2030, 2033, and 2035, respectively [4][5][6].

In Sweden, the impact of these changes is particularly significant due to the strong presence of major car manufacturers like Volvo, Volkswagen, and KIA, whose vehicles are among the most common on local roads. Their shift to fully electric models underscores Sweden's commitment to sustainable mobility and climate action. This aligns with broader European objectives and highlights Sweden's leadership in environmentally conscious transportation within the framework of the European Green Deal.

However, as the number of EVs increases, so does the demand for charging infrastructure. Forecasting the occupancy of EV charging stations—predicting how many chargers are occupied at a given time—is crucial for optimizing usage and improving user experience. This information aids city planners and companies in developing efficient charging networks that is easily accessible to EV drivers. Access to reliable real-time information and future predictions about charger availability can help drivers plan their routes more efficiently, reducing waiting times and ensuring they can find available chargers when needed. This not only promotes the use of diverse stations within a region, preventing excessive load on a single station but also reduces the anxiety associated with finding an available charging point, thus enhancing the overall adoption and satisfaction of using EVs. Moreover, efficient management of EV charging infrastructure is essential to accommodate the growing number of EVs. As the EV market continues to expand, ensuring the availability and reliability of charging stations is crucial for maintaining user satisfaction and promoting further adoption of EVs. Enhanced forecasting and data-driven strategies play an important role in addressing these challenges and supporting the transition to a more sustainable transportation system.

Considering the challenges and needs outlined above, this research project aims to develop a Deep Learning (DL) solution capable of predicting charging station occupancy by forecasting the ratio of occupied chargers to the total number of chargers available at a station. Specifically, the project focuses on developing a Spatio-Temporal Graph Convolutional Network (ST-GCN) that forecasts the occupancy of multiple nearby stations simultaneously. However, the model's training process is time-intensive for each charging station it needs to forecast. Additionally, when insufficient historical data is available for a charging station, the training may yield an inaccurate model. To address these challenges, this research proposes a transfer learning adapter, the *Bridged Attention Module (BAM)*, for efficient fine-tuning of the ST-GCN model, enabling the DL solution to scale easily to a large number of stations. By incorporating attention mechanisms, the model can be fine-tuned on multiple stations with limited historical data while still producing accurate results.

This research is conducted in collaboration with ChargeFinder [7], a Swedish website and mobile app that provides information on EV chargers and their availability. ChargeFinder aggregates data from various charging platforms and individual station providers to offer comprehensive and real-time information about charging station availability, pricing, and other relevant details [7]. This information is accessible to EV drivers through both the website and the mobile application. A key feature of ChargeFinder is its integration with the Mapbox Charge Finder API [8], which expands its database to include over 750000 charging stations in more than 70 countries worldwide.

However, this research project focuses exclusively on charging stations in Sweden, and an overview of the data is presented in Section 2. The data will be used to train and test the DL solution developed for this project. ChargeFinder aims to benefit from this research by potentially implementing the findings and methodologies into their website and app. The data from ChargeFinder plays a crucial role in the analysis and forecasting efforts presented in this thesis.

1.1 PROBLEM DEFINITION

The rapid growth of the EV industry, as discussed in Section 1, presents significant challenges in managing charging infrastructure to meet increasing demand. Accurate forecasting of EV charging station occupancy is essential for optimizing usage, reducing waiting times, and enhancing the user experience. Researchers have approached this problem in two primary ways:

- **Single-charging station forecasting:** Some studies focus on predicting the occupancy of individual charging stations using time series forecasting methods. These approaches analyze past usage patterns to estimate future availability at specific stations [9][10][11][12].
- **Multi-charging station forecasting:** Other studies expand this approach to forecast occupancy across entire *charging networks*, which consists of multiple stations in close proximity to each other [13][14]. In this scenario, the occupancy of one station can influence the occupancy of others within the network, as well as external events might affect multiple stations similarly. This broader perspective helps to understand and manage the availability of charging stations across a wider area. For example, if two stations are nearby and one consistently reaches full capacity before the other, it becomes possible to predict an increase in usage at the second station as the first approaches full capacity.

This thesis focuses on the multi-station prediction approach. By leveraging data from multiple charging stations, tha goal is to forecast occupancy across a broader area. Utilizing the interdependence between stations allows the DL model to predict usage patterns more accurately, thereby optimizing the overall efficiency of the charging infrastructure.

However, a critical challenge in forecasting occupancy across multiple charging stations and networks is the large number of stations in the territory. In Sweden alone, there are 7103 charging stations [7]. Training models that account for all these stations require substantial computational power and time. Additionally, for newly placed charging stations, only limited historical data is available, resulting in poor forecasting and generalization capabilities for a DL model.

To address these challenges and enhance the model's performance, this research explores the application of transfer learning—a ML technique where a model trained on one charging network is reused as the starting point for training on another network [15]. Since the task of predicting charging station occupancy is similar across different networks, using a model trained on one network as a base for another network allows for the transfer of valuable information, thereby improving the new model's accuracy. Transfer learning has proven highly effective in enhancing model performance with limited data across various domains, such as natural language processing [16] and computer vision [17].

This research project aims to improve the accuracy and efficiency of training and forecasting EV charging station occupancy by leveraging a novel adapter architecture. This approach will enable robust predictions and reliable charging infrastructure management, even with limited datasets.

1.2 GOAL

The goal of this research project is to develop and implement an ST-GCN as a reliable method for forecasting EV charging station occupancy. This model is designed to capture and analyze both spatial and temporal patterns, enabling accurate predictions by understanding the interdependence between stations. In parallel, the project includes analyzing data from ChargeFinder to uncover patterns in charging station occupancy. Additionally, a novel transfer learning adapter is introduced to improve the performance of the ST-GCN across diverse networks, even with limited historical data, while minimizing the number of trainable parameters. In summary, this research project focuses on three key areas:

- **Data Analysis:**

  This research involves a comprehensive analysis of the intermittent data provided by ChargeFinder, which cannot be directly used for time series prediction due to its irregular and intermittent nature. The data consists of sporadic records with significant gaps, reflecting real-world user queries rather than continuous observations. This study explores various data imputation techniques to fill the gaps and resample the dataset into a continuous dataset suitable for reliable forecasting. By experimenting with multiple imputation methods, the research aims to determine the most effective approach for creating realistic data that accurately represents charging station occupancy trends. In addition to data imputation, the direct utilization of intermittent data for forecasting was also investigated, which highlighted that data imputation is essential for achieving good predictive performance. The research also identifies and rectifies issues within the data, such as missing or unrealistic values, to ensure the dataset is well-prepared for time series prediction.

- **Forecasting:** Several methods for time series forecasting are employed and benchmarked to determine the most suitable approach for the data provided by ChargeFinder, as described in Sections 4.2 and 4.3.

  The core model developed in this research project to forecast the occupancy levels of multiple charging stations is the ST-GCN, as discussed in Section 1.1. The ST-GCN combines graph convolutional operations with recurrent layers layers to effectively capture and model both spatial and temporal patterns. It is designed to predict the occupancy of multiple stations simultaneously, taking into account the interactions between stations that influence each other's occupancy as well as the effects of external events that impact all stations across the network uniformly.

By experimenting with various approaches, the research aims to identify the most reliable methods for forecasting EV charging station occupancy. These include using both Long Short-Term Memory (LSTM) and Transformer as the recurrent layers and experimenting with using intermittent data as input to the Transformer model.

The models in this research project are designed to forecast up to 3 hours into the future. This timeframe strikes a balance between the challenges of making accurate long-term predictions and the limited usefulness of very short-term predictions that may not provide sufficient time for effective planning. Forecasting up to 3 hours into the future offers an optimal balance by providing drivers with valuable information at the start of their trip about where to charge their EV, while still ensuring a high level of prediction accuracy. Shorter forecasts, such as those under an hour, offer limited practical value since most drivers aren't concerned about charging decisions within the next 20 minutes. Instead, they need reliable information for the entire battery charge, which, on average, lasts around 3 hours and 45 minutes, as explained below. However, forecasting much further into the future increases uncertainty, making predictions less accurate as time progresses. Previous research has varied in the prediction timeframe, ranging from as short as 30 minutes to 2 hours [18][13][12], to as long as several days [19]. Additionally, the International Energy Agency (IEA) reports that the average range of a large, high-end EV is $360 - 380$ km, equating to roughly 3 hours and 45 minutes of highway driving [20]. Therefore, a 3-hour forecast provides a practical balance, offering occupancy predictions long enough to be useful for EV drivers without unnecessarily extending the prediction period. For more information regarding why exactly 3 hours was chosen please refer to chapter 5.1.2.

- **Transfer Learning:** The final objective of this research project is to streamline the training process for a large number of charging networks using transfer learning techniques. This approach involves pre-training the ST-GCN on one network of charging stations and subsequently fine-tuning it on another, utilizing 1%, 5%, 20%, and 50% of the target data. Various transfer learning strategies are tested and benchmarked, including BAM, a novel adapter architecture developed specifically for this project. The goal is to identify the most effective transfer learning techniques that optimize performance relative to the number of trainable parameters, thereby reducing computational cost without compromising performance.

Ultimately, the project aims to enhance EV charging infrastructure by providing accurate occupancy forecasts and optimizing route planning. This will improve the efficiency and responsiveness of the charging infrastructure, supporting the adoption of EVs and promoting sustainable transportation solutions.

1.3 RESEARCH QUESTIONS

1. **Data:** What specific data cleaning, transformation, and imputation techniques can be applied to ChargeFinder's dataset for addressing issues such as missing values, outliers, and data inconsistencies, thereby enhancing the accuracy of forecasting the EV charging station occupancy?
2. **Forecasting:** How can we design and optimize a forecasting model specifically for the ChargeFinder dataset to minimize Mean Squared Error (MSE) when predicting EV charging station occupancy for the next three hours?
3. **Transfer Learning:** How can we implement and optimize a transfer learning technique that minimizes the number of trainable parameters while maintaining competitive MSE performance for predicting EV charging station occupancy?

2 DATA

This thesis utilizes data from ChargeFinder [7], a digital platform that provides real-time status updates on charging stations. It is important to note that ChargeFinder does not own these stations; rather, it aggregates information from various providers across Europe and America. As a result, the charging stations considered in this thesis are managed by different providers, leading to variations in pricing, payment options, charging power, charging port types, and opening hours.

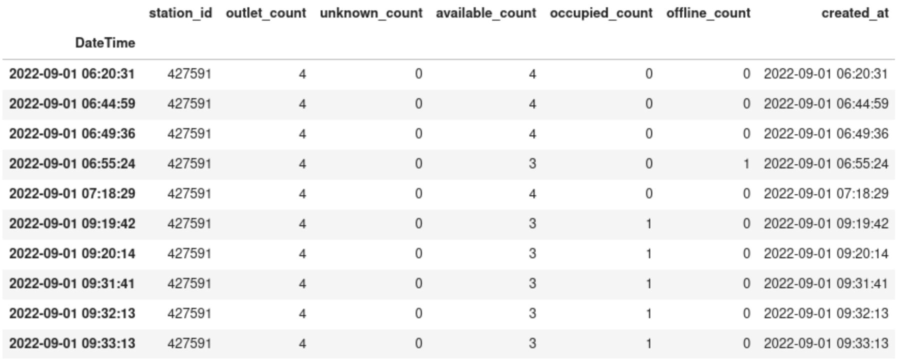
The data utilized in this thesis consists of 29 charging stations located across three cities in Sweden: Värnamo, Varberg and Malmö. The stations are listed in Table 1, along with the corresponding number of data points for each station. Notably, there is a significant disparity between the station with the most data points, IONITY, and the one with the fewest, Hälsingland_19. When comparing the average `occupied_count` in Figure 2, and compare it to Table 1, it becomes evident that stations with the highest average `occupied_count` also tend to have the most data points.

2.1 FEATURE OVERVIEW

Figure 1 presents an extract of the dataset from McDonald's charging station in Värnamo. Each dataset consists of seven attributes describing EV charging, as outlined in Table 2. The `station_id` column provides a unique identifier for each station, which can be cross-referenced with a lookup table to obtain the real station name and lo-

Värnamo		Varberg		Malmö	
Station name	**Data**	**Station name**	**Data**	**Station name**	**Data**
UFC	46740	Varberg_UFC	26691	Emporia_Plan_3	9617
IONITY	71024	IONITY_Varberg	50527	P_Huset_Plan_3	4845
Jureskogs_Vattenfall	32123	Toveks_Bil	12727	P_hus_Allé	4940
McDonalds	30430	Varberg_Supercharger	31009	OKQ8_Svansjögatan	10060
Holmgrens	36887	Recharge_ST1	14444	P_hus_Malmömässan	5148
BurgerKing	15460	Lunds_Bil	12576	P_hus_Södervärn	3823
OKQ8	12788	Circle_K_Varberg_Nord	9746	P_hus_Stadion	3247
ICA	7525	Circle_K_Varberg_Nord_Trucks	8544	OKQ8_Kvartettgatan	3938
Jureskogs_Recharge	1292	Vagnvägen_Mantum	3508	Hälsingland_19	439
		NIO	4326		
		OKQ8_Monarkvägen	638		

Table 1: All stations considered in this project, alongside the amount of corresponding data points attributed to them. The names are the same as in the lookup table provided by ChargeFinder.



DateTime	station_id	outlet_count	unknown_count	available_count	occupied_count	offline_count	created_at
2022-09-01 06:20:31	427591	4	0	4	0	0	2022-09-01 06:20:31
2022-09-01 06:44:59	427591	4	0	4	0	0	2022-09-01 06:44:59
2022-09-01 06:49:36	427591	4	0	4	0	0	2022-09-01 06:49:36
2022-09-01 06:55:24	427591	4	0	3	0	1	2022-09-01 06:55:24
2022-09-01 07:18:29	427591	4	0	4	0	0	2022-09-01 07:18:29
2022-09-01 09:19:42	427591	4	0	3	1	0	2022-09-01 09:19:42
2022-09-01 09:20:14	427591	4	0	3	1	0	2022-09-01 09:20:14
2022-09-01 09:31:41	427591	4	0	3	1	0	2022-09-01 09:31:41
2022-09-01 09:32:13	427591	4	0	3	1	0	2022-09-01 09:32:13
2022-09-01 09:33:13	427591	4	0	3	1	0	2022-09-01 09:33:13

Figure 1: An extract of the dataset corresponding to McDonald's charging station in Värnamo.

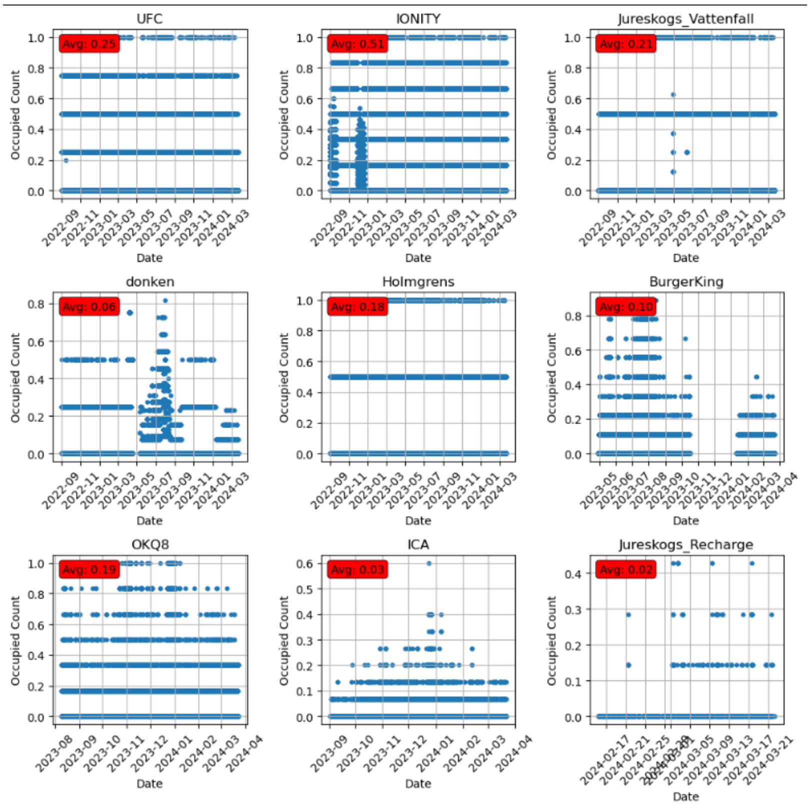


Figure 2: `occupied_count` for all stations in Värnamo.

Column	Non-Null Count	Dtype
station_id	163, 618	int64
outlet_count	163, 618	int64
unknown_count	163, 618	int64
available_count	163, 618	int64
occupied_count	163, 618	int64
offline_count	163, 618	int64
created_at	163, 618	object

Table 2: Data columns information for the IONITY dataset.

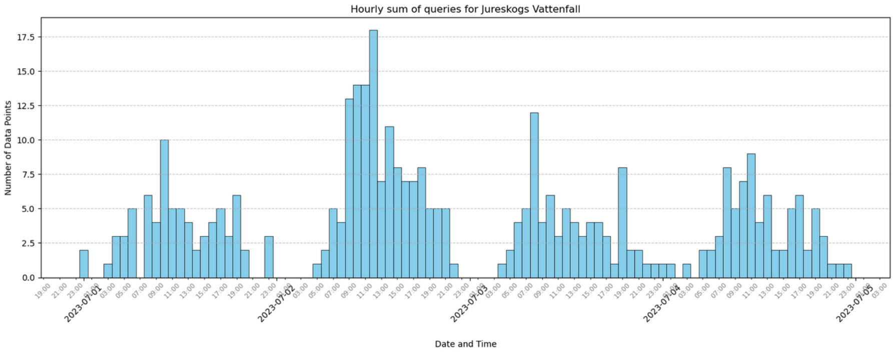


Figure 3: An extract consisting of four days from the jureskogs_vattenfall dataset and their hourly frequency of data points.

cation. Additionally, the dataset includes columns such as `outlet_count`, which indicates the number of charging outlets at each station, along with `unknown_count`, `available_count`, `occupied_count`, and `offline_count`, representing the counts of charging stations in different states: unknown, available, occupied, and offline, respectively. These features are crucial for understanding the current status and utilization of each charging station.

The `created_at` column contains timestamp information, recording the date and time when a user queries the ChargeFinder website. However, an examination of the `created_at` column in Table 2 reveals that the data is intermittent. This intermittent nature is further illustrated in Figure 3, which bins the data points into 1-hour intervals and displays them in a density plot. The plot shows that the data is grouped into four distinct clusters over four consecutive days, with significant gaps during nighttime due to minimal querying activity.

2.2 INTERMITTENT DATA

The ChargeFinder dataset consists of intermittently recorded data, collected based on user queries rather than continuous monitoring. This results in gaps and irregular time intervals between data points, making it challenging to directly apply standard time series forecasting techniques. Due to the irregular intervals and gaps, the models may struggle to identify meaningful patterns or trends, leading to reduced accuracy in occupancy predictions. Moreover, the data sparsity can cause the models to overfit to the limited available observations or fail to generalize to new scenarios. Therefore, this research attempts to resample the data using different imputation techniques to create a more consistent and robust dataset for reliable model training. By implementing methods to manage irregular intervals and gaps, this thesis aims to improve the accuracy of forecasting models and provide a basis for determining the best data handling practices for the ChargeFinder dataset. This approach directly addresses the first research question, which seeks to identify effective data cleaning, transformation, and imputation techniques to handle the intermittent nature of the ChargeFinder dataset.

2.3 DATA TRENDS AND ANOMALIES

To visually explore potential trends within the dataset, a plot of the daily averages across the entire temporal span is generated for the five stations in Värnamo, focusing on the `occupied_count` feature. This analysis, as depicted in Figure 4, is performed using the *scikit-learn* library [21].

The analysis of the trend across the five charging stations reveals pronounced seasonal patterns, with consistent peaks during specific periods. These peaks occur at all stations, but are particularly evident in the purple curve during the intervals of December 2022 to January 2023, March to May 2023, July to September 2023, and again from December 2023 to January 2024. This pattern underscores the dynamic nature of the stations' occupancy rates, reflecting a clear seasonal rhythm and periods of heightened demand. However, as seen in Figure 4, some data is missing for the `donken` station (represented in red) from late April 2023 to early May 2023. A direct examination of the `donken` dataset confirms that no data points were logged from $2023-04-1804:55:20$ to $2023-05-0909:09:37$. This extended period of missing data may be due to a fault with the charger provider, which is beyond the scope of this thesis. However, this issue affects the forecasting ST-GCN during this period due to the multi-station forecasting approach used relies on data from all stations simultaneously. To mitigate this issue, this timeframe was ex-

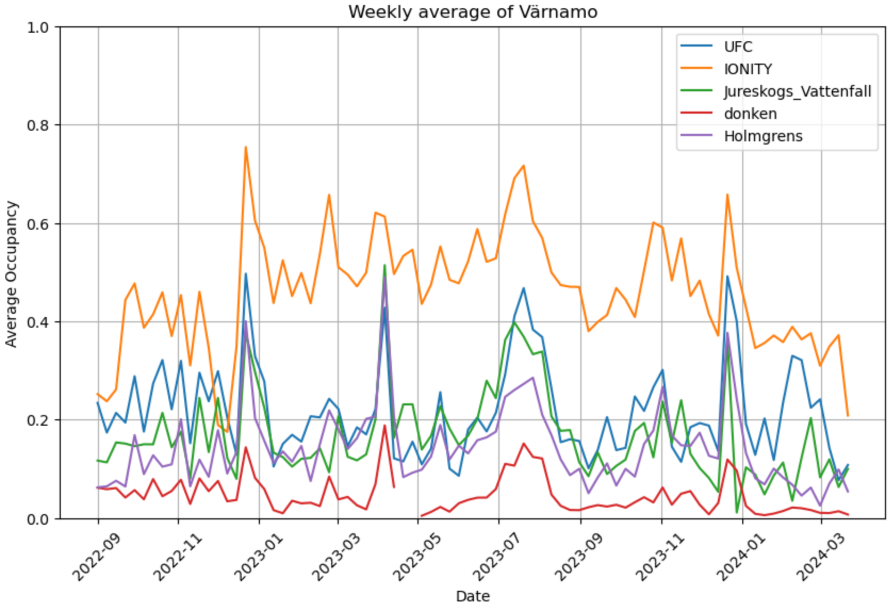


Figure 4: Visualization of the trend for the five charging stations used in Värnamo.

cluded from the tests, ensuring that the models do not encounter this portion of the datasets during the experiments.

Additionally, a significant drop in occupancy is observed at the IONITY charging station in Värnamo during September and early December 2022. These drops coincide with a sudden and temporary increase in the number of chargers, as shown in Figure 5. This discrepancy suggests that the data may be faulty, and the total number of chargers during this time remains stable at six. This would explain the temporary decrease in occupancy during this period in the IONITY dataset since the `occupied_count` feature is divided by the `outlet_count` feature during normalization, resulting in a higher denominator than the actual number of existing chargers. To address this, the `outlet_count` for the IONITY charging station should be limited at six.

Furthermore, the dataset varies in granularity based on user activity and geographic location. High-traffic areas with frequent EV usage and charging stations have robust data coverage, while less populated regions and providers may exhibit significant data sparsity. This variability necessitates careful data handling and pre-processing to ensure the reliability and accuracy of the forecasting models used in this thesis. As a result, data pre-processing steps such as handling missing values, normalizing feature distributions, and segmenting data into meaningful time intervals will be performed to optimize model training. Addressing these challenges aims to improve the prediction models' performance and enhance the understanding of charging station usage patterns, ultimately contributing to more effi-

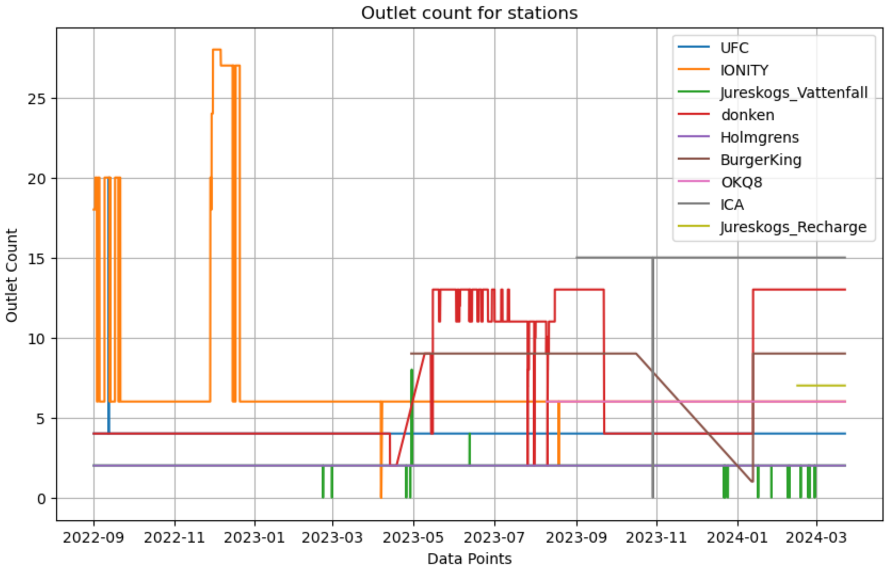


Figure 5: The total number of chargers/station over time in Värnamo.

cient and reliable forecasts. More details regarding the pre-processing stage are described in Section 4.1.

3 RELATED WORKS

This chapter reviews several related works, beginning with an exploration of data imputation techniques. It then discusses forecasting models, covering both ML and DL approaches. Following this, the chapter examines the application of transfer learning and concludes with a discussion of evaluation methods. These studies offer innovative strategies and insights that are highly relevant and beneficial to this research project.

3.1 DATA IMPUTATIONS

To convert intermittent time series data, as described in Chapter 2, and illustrated in Figure 3, into a continuous format, imputation techniques are essential. These techniques are typically classified into two categories: univariate and multivariate imputation [22].

3.1.1 *Univariate*

Univariate imputation involves filling in missing values within a single variable using only information from that variable. In a project with a similar intermittent dataset [23], temporally-based univariate strategies such as Last Observation Carried Forward (LOCF) and Next Observation Carried Backwards (NOCB) were employed. LOCF fills in gaps using the last observed value before the missing data point, while NOCB uses the next observed value after the missing data point. Both methods operate under the assumption that the preceding or succeeding measurement is a reasonable estimate for the missing value.

Mean, median, and mode imputation are straightforward methods for handling missing data by replacing missing values with a single statistic: the mean, median, or mode of the observed values [24]. Mean imputation uses the average of the available data, which is suitable for numerical variables. Median imputation replaces missing values with the middle value, making it robust against outliers and ideal for skewed distributions. Mode imputation fills gaps with the most frequent value, typically used for categorical variables. While these methods are easy to implement, they have limitations, such as ignoring relationships between variables and reducing data variability, which can lead to biased estimates.

3.1.2 *Multivariate*

Multivariate imputation considers the relationships and dependencies between multiple variables to provide more accurate imputations for missing data [22]. One commonly used technique is k-nearest neighbour (k-NN), which can be applied as both a univariate and multivariate method. For univariate imputation [25], k-NN fills missing values in a single variable by considering the values of the nearest neighbours in that variable. For multivariate imputation, however, k-NN estimates and replaces missing values by leveraging similarities across multiple variables, utilizing the relationships between different data points to produce more informed estimates [11].

Another widely used multivariate imputation method is Multivariate Imputation by Chained Equations (MICE) [26]. MICE assumes that the missing data is "missing at random" and imputes missing values by considering the relationships among different variables in the dataset. The process begins with initial guesses for the missing values, often using the mean or median. Then, models are built for each variable with missing data, using other variables to predict the missing values. This iterative process is repeated multiple times, refining the predictions with each iteration until they stabilize. MICE generates several versions of the dataset to account for uncertainty, ultimately combining these to produce the final estimates.

3.1.3 *Summary*

The data imputation section outlines various methods for handling missing data, categorized into univariate and multivariate techniques. Univariate methods include LOCF and NOCB, which fill missing values based on the nearest observed data points. Simpler methods like mean, median, and mode imputation replace all missing values with the respective statistics of the observed data. On the other hand, multivariate methods such as k-NN and MICE leverage relationships between multiple variables to estimate missing values, providing more robust and accurate imputations.

However, not all of these methods are suitable for resampling an intermittent time series into a continuous one. Mean, median, and mode imputation fill all missing values with the same value, which would lead to a dataset that is not useful when converting to a continuous time series. MICE is also inappropriate because it assumes the data is "missing at random," which does not apply to the dataset used in this thesis. In this case, data points are logged only when users query the ChargeFinder website, meaning the absence of data points is driven by external factors not captured within the dataset. Consequently, MICE and mean/median/mode imputation are unsuitable for resampling the dataset from intermittent to continuous.

The methods deemed appropriate and will be explored in this project are LOCF, NOCB, and k-NN.

3.2 MACHINE LEARNING

The following subsections explore various ML methodologies commonly employed in time-series forecasting. Moving Average (MA), Auto Regression (AR), and AutoRegressive Integrated Moving Average (ARIMA) are fundamental linear models utilized in time series prediction, as demonstrated in numerous research studies [27][28][29].

3.2.1 *MA*

The MA model, introduced by Herman Wold in 1938 [30], is one of the earliest time series models developed. It is widely used in various fields, such as finance for stock price forecasting, economic indicator analysis, and meteorology for predicting weather patterns [31][32][33].

The strengths of the MA model include its simplicity, ease of implementation, and interpretability. It effectively smooths out short-term fluctuations while highlighting underlying trends in the data. By focusing on prediction errors, the MA model filters out noise, providing clearer insights into the true signal of the time series [34][35].

However, the MA model assumes that errors are independently and identically distributed, an assumption that may not hold true with real-world data. Additionally, the MA model only utilizes univariate datasets and captures linear relationships, making it less effective for complex time series data with nonlinear patterns. Its effectiveness also diminishes with longer forecast horizons, as it relies heavily on recent errors, which may not adequately capture long-term trends [36][37][38].

3.2.2 *AR*

Alongside the MA model, the AR model, introduced by Yule in the 1920s [39], is also one of the earliest time series models. It is widely used in economics, finance, and other fields for modeling and forecasting, similar to the applications of the MA model described in the previous subsection [40][41].

The AR model's benefits include its simplicity and effectiveness in capturing linear dependencies within time series data. It is straightforward to understand and implement. However, like the MA model, the AR model assumes that the time series is stationary, focuses solely on univariate data, and is limited to capturing linear relationships [42][43].

3.2.3 *ARIMA*

To address the limitations of MA and AR models in handling non-stationary data, Box and Jenkins developed the ARIMA model in the 1970s [44]. This model combines both MA and AR components and effectively manages non-stationary data by incorporating differencing, making it a versatile tool for various applications [45].

The ARIMA model is particularly notable for its ability to handle both stationary and non-stationary time series data, offering greater flexibility compared to the MA and AR models. Its integration of AR and MA components makes it a more comprehensive tool for time series analysis. However, ARIMA models require careful selection of order parameters (p, d, q) and can be more computationally intensive than simpler models. Additionally, while ARIMA addresses the need for data stationarity, it still relies solely on univariate data and may struggle with datasets that exhibit complex seasonal patterns or nonlinearities [46][47].

3.3 DEEP LEARNING

Advanced DL models such as LSTM [48], Transformer [49], Graph Convolutional Network (GCN) [50], and ST-GCN [14] offer sophisticated techniques for time series prediction, demonstrating superior performance to ML algorithms in handling complex and high-dimensional data. This section provides a comprehensive comparison of these models, highlighting their strengths, limitations, and appropriate application scenarios.

3.3.1 *LSTM*

In the realm of time series forecasting, the limitations of traditional ML models — such as their inability to capture long-term dependencies, handle high-dimensional data effectively, and model complex nonlinear relationships — have led researchers to increasingly focus on DL methods over the past decade. LSTM networks, in particular, have become a cornerstone in this field due to their effectiveness in handling time series data [48]. These networks excel in time series prediction by leveraging an internal memory mechanism that updates iteratively with each step in the time series, as highlighted in the following papers [12][18].

Originating from the seminal paper in 1997 [51], LSTM networks have been widely adopted across various domains. For instance, a study [52] shows that LSTM networks are used in 25 out of 34 studies focusing on stock market forecasting. This underscores the effectiveness of LSTM models in time series prediction, demonstrating their capability to capture complex temporal dependencies in data.

An LSTM network is composed of LSTM cells, each equipped with gates that control the flow of information and preserve long-term dependencies. In networks with multiple layers, these LSTM cells are stacked, enabling the network to capture intricate patterns and hierarchical representations within the data. Additionally, LSTM networks can be organized into different architectures based on how they process input data. The main LSTM architectures include *many-to-one*, *many-to-many*, and *one-to-many*, each tailored to specific applications.

- In the *many-to-one* architecture, the model processes a sequence of inputs to generate a single output. This setup is advantageous for tasks like sentiment analysis, where a sequence of words is analyzed to predict overall sentiment [53], or stock price prediction [54], where historical data is used to forecast the next day's price. This architecture was notably used in the paper that introduced the forget gate to the LSTM [55]. Additionally, speech recognition tasks, where an audio sequence is identified as a spoken word, also benefit from this architecture [56].

- The *many-to-many* architecture [51] can be either synchronous or asynchronous. In synchronous many-to-many, each input in the sequence corresponds to an output, such as in machine translation or video frame labeling [57][58], where each frame requires a label. Asynchronous many-to-many handles cases where the input and output sequences differ in length, such as text summarization, where concise summaries are generated from longer documents, and question answering, where detailed responses are produced based on the context [59].

- In the *one-to-many* architecture, a single input leads to a sequence of outputs. This is ideal for tasks like image captioning [60], where descriptive text is generated from an image, and music generation [61], where a sequence of musical notes is created from an initial note.

These LSTM architectures provide tailored solutions for various sequence modeling tasks, leveraging LSTMs' strengths in capturing temporal dependencies and handling complex sequence data.

LSTM networks offer several notable strengths. Firstly, they are specifically designed to overcome the vanishing gradient problem common in earlier DL methods like Recurrent Neural Networks (RNN), enabling them to effectively capture long-term dependencies in time series data [62][63]. Secondly, the nonlinear activation functions within LSTMs allow them to model complex, nonlinear relationships in the data [64]. Lastly, LSTMs can be applied to various types of time series data, including univariate and multivariate datasets, providing flexibility in their usage [65]. These advantages address the limitations of models like MA, AR, and ARIMA, but they also introduce new challenges.

LSTMs are computationally intensive and require significant resources for training, which can be a drawback for large datasets or real-time applications [66]. Additionally, the performance of LSTM models is highly sensitive to the choice of hyperparameters, necessitating careful tuning and validation [67]. Furthermore, LSTMs have limitations in capturing contextual information within a sequence, which leads to the development and use of the Transformer model.

3.3.2 *Transformer*

In 2017, a groundbreaking paper introduced the Transformer neural network architecture [68], which demonstrated exceptional performance in natural language processing tasks. Due to the similarities between time series prediction and natural language modeling, the Transformer architecture has also shown promising results in the realm of time series forecasting [69]. Transformers operate using a sequence-to-sequence generation approach with an encoder-decoder architecture. The encoder processes input data to extract meaningful representations, while the decoder generates output sequences based on these representations and previous predictions.

Unlike the recurrent-based approach of LSTMs, the Transformer architecture relies entirely on attention mechanisms in both its encoder and decoder components to process input data [70]. This design allows the model to dynamically focus on relevant parts of the input when making predictions, significantly enhancing its ability to understand context and generate accurate forecasts. Additionally, the Transformer incorporates a multi-head attention mechanism, enabling it to process different aspects of the information in parallel. This self-attention process handles all items in a sequence concurrently, offering a novel approach to sequential data analysis.

When it comes to strengths and weaknesses, the Transformer architecture offers numerous advantages. The self-attention mechanism allows Transformers to consider the entire sequence of data simultaneously, providing a global context that enhances their ability to capture long-range dependencies effectively [71]. Unlike LSTMs, which process data sequentially, Transformers can process multiple data points in parallel. This parallel processing capability significantly reduces training time and improves computational efficiency, making Transformers more scalable for large datasets [72]. Additionally, Transformers demonstrate remarkable flexibility across various tasks beyond time series forecasting, including natural language processing and computer vision, showcasing their adaptability to different types of data and problem domains [73].

However, Transformers also have some notable weaknesses. Despite their efficiency in parallel processing, they are computationally expensive due to high memory requirements, particularly when dealing with very long sequences or large-scale models [74]. Moreover, Transformers typically require large amounts of training data to achieve optimal performance, which can be a limitation in scenarios where data is scarce or costly to obtain [75].

3.3.2.1 *Positional Encoding*

In Transformer models, especially when applied to time series data, positional encoding is crucial for conveying the sequence order to the model. Positional encoding is used by adding a unique positional vector to each input token, allowing the model to understand the order and position of elements in a sequence. A widely used method is Absolute Positional Encoding, as detailed in [68][76][49]. This approach integrates absolute positional information into the model using specific mathematical equations, as shown in Equations 12 and 13, enabling the model to perceive and process the sequence order.

Relative Positional Encoding [77] shifts the focus from absolute positions to the relative distances between tokens. Instead of encoding the absolute position of each token, this method emphasizes the relative positioning between them. For instance, for a token at position i attending to another at position j, the relative position is defined as $k = j - i$. The model learns embeddings for these relative positions, allowing it to better understand and leverage the positional relationships between tokens based on their relative distances. This approach enhances the model's ability to generalize across different sequence lengths and structures by providing a more nuanced understanding of how tokens influence each other.

Another method is Fractional Positional Encoding [78], which improves the temporal resolution in video-to-text models. The authors of [78] addressed the challenge of unsynchronized audio and video features by introducing fractional positional encoding. Rather than resampling the data, this method encodes the time differences between audio and video features into the tokens before feeding them into the Transformer. This approach enables the model to better align and interpret audio-visual data, leading to more accurate video-to-text translations.

3.3.3 *GCN*

Graph Neural Networks (GNN) extend neural network capabilities to graph-structured data, enabling the modeling of complex relationships and interactions between entities within a graph [79]. These networks are particularly effective for tasks involving nodes and edges, such as node classification and link prediction.

GCNs, a specialized form of GNNs adapted from Convolutional Neural Networks (CNN), are designed to analyze data with spatial relationships [50]. Unlike CNNs, which apply convolutional layers to image data, GCNs extend the concept of convolution to graph-structured data, allowing for the processing of spatial relationships in non-Euclidean domains.

In the original 2016 paper introducing GCNs [50], the authors demonstrated the effectiveness of GCNs on several citation network datasets, where nodes represent documents and edges represent citation links. The GCN model significantly outperforms other state-of-the-art methods for node classification in this semi-supervised setting, highlighting its potential for a wide range of applications in graph-structured data analysis.

GCNs offer several advantages. They excel at capturing local neighborhood information in graph-structured data, which is essential for tasks like node classification and link prediction. Their ability to operate on non-Euclidean data allows them to handle complex relationships that traditional CNNs cannot [80]. Additionally, GCNs have shown superior performance in semi-supervised learning settings, as they can leverage both labeled and unlabeled data [81].

However, GCNs also have some limitations. They can be computationally expensive, particularly for large graphs, due to the recursive nature of the convolution operations. The effectiveness of GCNs can diminish as the network depth increases, leading to issues such as over-smoothing, where node representations become indistinguishable from one another [82]. Furthermore, GCNs require careful tuning of hyperparameters and may struggle to scale effectively to extremely large graphs [83].

3.3.4 *ST-GCN*

ST-GCNs are advanced models designed to capture both spatial and temporal dependencies in the data by incorporating two components: one for the spatial aspect and one for the time series aspect. They are particularly effective in applications where time series data has an inherent spatial relationship, such as traffic prediction [84].

In [85], a novel method for recognizing hand gestures by utilizing skeleton data of hand joints is presented. The study employs an ST-GCN to capture spatial relationships between joints and temporal dynamics of gestures. The ST-GCN model applies graph convolutions to the hand skeleton in each frame for spatial analysis and temporal convolutions across frames to understand movement patterns over time, resulting in improved accuracy and efficiency in gesture recognition tasks.

A particularly relevant example of ST-GCNs applied to time series forecasting is found in [14], which focuses on forecasting occupancy at charging stations. This ST-GCN model employs a Gated Recurrent Unit (GRU) to capture temporal dynamics and a GCN to address spatial relationships. The graph is modeled as a city road network, with nodes representing both charging stations and intersections.

In another study, [13], a GCN is combined with an Informer instead of a GRU. The Informer [86], a Transformer-based method, processes the entire sequence simultaneously and incorporates auxiliary data like weather information for more accurate occupancy forecasts. Unlike the full road network model used in [14], [13] simplifies the network by considering only charging stations and the distances between them.

ST-GCNs offer multiple advantages. They excel at capturing both spatial and temporal dependencies in data, making them ideal for spatio-temporal forecasting tasks. By integrating GCNs to manage spatial relationships and temporal models like GRUs, LSTMs, or Transformers for temporal dependencies, ST-GCNs provide a robust framework for complex forecasting tasks [87]. Additionally, their ability to model intricate networks, such as city road networks, enhances their applicability in urban planning and transportation systems [84].

However, ST-GCNs can be computationally intensive, requiring significant resources for both training and inference, especially when handling large-scale spatio-temporal data [88]. The integration of different model components, such as GCNs and temporal models, also necessitates careful tuning of hyperparameters and architecture design, which can be challenging and time-consuming [89]. Moreover, the effectiveness of ST-GCNs is heavily dependent on the quality and completeness of the spatial and temporal data available, as any missing or inaccurate data can significantly impact the model's performance [90].

3.3.5 *Summary*

The choice of a time series forecasting model significantly impacts the accuracy and reliability of predictions. Table 3 provides a comprehensive overview of the strengths and weaknesses of the models discussed, which supports the selection of the forecasting model used in this thesis.

Traditional ML models like MA, AR, and ARIMA offer simplicity and effectiveness in certain contexts but are limited in their ability to handle complex, nonlinear patterns, non-stationary data, and long-term dependencies. These models are univariate and rely on linear relationships, which restricts their applicability in real-world scenarios where data often exhibit nonlinearities and seasonal behaviors.

In contrast, advanced DL models provide significant improvements in capturing complex temporal dependencies and nonlinear relationships. Compared to traditional ML models, DL models like LSTMs and Transformers are better suited for handling large datasets and learning intricate patterns within the data. However, these models are computationally intensive, require extensive data for training, and their performance is highly sensitive to hyperparameter tuning.

Graph-based models like GCNs offer a robust method for capturing spatial dependencies in graph-structured data. GCNs excel in semi-supervised learning and operating on non-Euclidean data but face challenges with computational expense and over-smoothing in deeper networks.

Considering these factors, the decision was made to utilize an ST-GCN that integrates a GCN for spatial dependencies and a Transformer for temporal analysis. This hybrid model leverages the strengths of both components: the GCN effectively captures local neighborhood information, while the Transformer excels in modeling long-term temporal dependencies and global context within sequences. Additionally, the Transformer's parallel processing capability improves inference and training times. For the results of this forecasting model and comparisons with other baselines, please refer to Section 5.2.

Model	Strengths	Weaknesses
MA	Simple. Effective in smoothing short-term fluctuations. Noise reduction.	Assumes independence of errors. Limited to linear relationships. Ineffective for long forecast horizons. Only univariate.
AR	Simple. Effective in modeling stationary time series data.	Assumes stationary time series. Limited to linear relationships. Ineffective for long forecast horizons. Only univariate.
ARIMA	Handle both stationary and non-stationary time series data. Flexible with both AR and MA components.	Complexity in parameter selection. Computationally intensive. Limited performance with seasonal patterns. Limited to linear relationships. Only univariate.
LSTM	Captures long-term dependencies. Handles nonlinear relationships. Solves the Vanishing Gradient problem. Both univariate and multivariate.	Computationally intensive. Requires large datasets. Sensitive to hyperparameters. No contextual information in a sequence.
Transformer	Captures long-term dependencies. Global contextual understanding in a sequence. Parallel processing. Flexible domain applications.	Computationally intensive. Requires large datasets. Memory intensive.
GCN	Captures local neighborhood information in graph-structured data. Operates on non-Euclidean data. Superior performance in semi-supervised learning.	Computationally expensive for large graphs. Over-smoothing with deeper networks. Requires careful hyperparameter tuning.
ST-GCN	Integrates and captures spatial and temporal dependencies. Ideal for spatio-temporal forecasting. Effective in complex network modeling.	Computationally intensive. Requires careful tuning. Dependent on data quality.

Table 3: Common time series forecasting models: summary of strengths and weaknesses.

3.4 TRANSFER LEARNING

Transfer learning is a machine learning technique in which a pre-trained model developed for one task is reused as the starting point for a model on a different, but related, task [91]. This approach is particularly beneficial when dealing with limited data, as it allows the model to generalize better by leveraging knowledge from related tasks or domains. Transfer learning is widely adopted in various fields, including computer vision, natural language processing, and, more recently, time series forecasting and spatio-temporal modeling.

Over the years, transfer learning has advanced significantlly, driven by contributions from numerous researchers. A survey conducted by leading researchers in 1995 presented some of the most innovative research approaches of that time, which have since evolved into what is now known as transfer learning [92]. The survey focused on exploring the utility and feasibility of computer programs that can "learn how to learn", both from a practical and a theoretical perspective, and it had a profound influence on subsequent research in the field.

More recent papers and surveys [93][94][95][96][97] have introduced various methodologies to enhance the efficiency and effectiveness of transfer learning. Key approaches to transfer learning include:

1. **Fine-tuning the Last Layer**: This approach involves updating only the parameters of the last layer while keeping the rest of the network frozen. It is computationally efficient and reduces the risk of overfitting, as demonstrated in the study by Yosinski et al. [93]

2. **Fine-tuning All Layers**: In this method, all layers of the pre-trained model are updated during the training process on the new task. While this can lead to improved performance, it is computationally expensive and may require more careful hyperparameter tuning to avoid overfitting, as discussed by Howard and Ruder [94].

3. **Using Adapters**: Adapters are lightweight modules inserted into selected positions of the pre-trained model. These adapters are fine-tuned while the original model's weights remain frozen. This approach, detailed by Houlsby et al., allows for efficient transfer learning with minimal additional parameters [95][96].

4. **Hybrid Approaches**: Combining different techniques, such as partially freezing layers or using a mix of fine-tuning and adapters, can optimize performance and efficiency, as described by Ruder et al. [97].

These methodologies offer a variety of strategies to effectively leverage pre-trained models for new tasks, each balancing trade-offs between computational cost, risk of overfitting, and potential performance gains.

3.4.1 *Adapters*

The authors of [95] introduced a novel approach to transfer learning by inserting lightweight modules, known as adapters, into each layer of a frozen pre-trained neural network. These adapters enable the model to be fine-tuned for specific tasks without altering the original parameters of the pre-trained model. The adapter module they developed consists of an efficient Multi-Layer Perceptron (MLP) network with two primary components: a downscaling block with a nonlinear activation function, followed by an upscaling block. This bottleneck structure regularizes the data, helping to mitigate overfitting, particularly in tasks with limited data. By reducing the number of trainable parameters compared to full model fine-tuning, this approach enhances both computational efficiency and effectiveness.

This MLP adapter module was first applied to Transformer architectures in Large Language Models (LLM), leveraging the layered structure of Transformers by inserting adapters between layers to efficiently manage the flow of information. This method preserves the valuable pre-trained knowledge while adapting it to new tasks with minimal computational overhead. The success of adapters in Transformers has led to their exploration in other neural network architectures, such as CNNs and RNNs. Researchers have demonstrated that similar benefits can be achieved in these architectures, broadening the applicability of adapters across various domains [98] [99] [100].

Building on the foundation of adapters, a paper titled [96] introduced a new variation called the "Tiny-Attention Adapter." Instead of using a simple bottleneck structure, this approach integrates a tiny multi-head attention layer within the adapter module, specifically applied in the LLM domain. The authors argue that this *tiny-attention* mechanism more effectively leverages contextual information than traditional, parameter-heavy methods. By focusing on the relevance of context, the Tiny-Attention Adapter achieves equivalent performance with minimal parameter updates compared to fine-tuning all parameters, thereby enhancing the model's adaptability and efficiency.

In their implementation, the Tiny-Attention Adapter is placed after the multi-head attention layer, rather than the feedforward layer, in the encoder. Additionally, the bottleneck size of the adapter is set to one, minimizing the number of parameters and justifying the name "tiny." However, while the paper claims that the adapter's weights constitute only 0.05% of the total model parameters, it is important to note that the decoder is also fine-tuned when fine-tuning the adapter on the target task. This means that although the adapter itself represents only 0.05% of the model's parameters, the actual number of parameters being retrained is significantly higher.

3.5 ANALYSIS AND OPTIMIZATION

The following subsections will focus on evaluating the model's results and performance. A thorough analysis of prediction outcomes is essential for refining forecasting models, ultimately improving their accuracy and efficiency.

3.5.1 *Evaluation metrics*

Evaluation metrics such as MSE are commonly employed in regression problems to provide comprehensive insights into a model's predictive performance [101][102][48]. MSE measures the average squared difference between the actual and predicted values, quantifying the magnitude of prediction errors. In addition to MSE, the standard deviation σ is a crucial metric for evaluating the dispersion of prediction errors. The standard deviation helps to understand the variability in model predictions, offering insights into the consistency and reliability of the model's performance. A lower standard deviation indicates that the predictions are closely clustered around the mean error, suggesting a more consistent model. The equations for these metrics are as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}, \tag{2}$$

where n is the total number of observations or data points in the dataset, Y_i represents the label for the i^{th} observation, and $\hat{Y}_i$ represents the predicted value for the i^{th} observation, x_i represents the i^{th} value of whatever series of numbers the standard deviation is calculated on, and μ is the average.

In this thesis, MSE will be used as the primary metric for comparing models. To ensure robustness, the models will be trained and evaluated across multiple seeds, with the average MSE from all runs used for comparison. Additionally, the standard deviation of the MSE values will be calculated to assess the consistency and reliability of the model's performance. Comparing standard deviations is crucial as it helps in understanding the variability in the models' predictions. A lower standard deviation indicates that the model's predictions are more consistent across different runs, suggesting greater reliability and stability in the model's performance.

3.5.2 *Hyperparameter Optimization*

There are several methods for optimizing hyperparameters, with one of the simplest approaches being grid search. Grid search is an exhaustive optimization algorithm that systematically explores a wide range of parameter combinations, using cross-validation to identify the most effective set of parameters for a given model [103][104][105].

Another popular method for hyperparameter optimization is random search. Unlike grid search, which evaluates all possible combinations systematically, random search selects random combinations of parameters to evaluate. This approach can be more efficient, especially in large search spaces, as it often identifies a good set of parameters more quickly, thereby reducing computational expense [106][107].

To fine-tune the model hyperparameters in this research, Optuna will be utilized, an open-source optimization framework in Python [108]. Optuna employs an efficient sampling algorithm called Tree-structured Parzen Estimator (TPE) [109], which models the distribution of hyperparameters based on past trials and focuses on the more promising regions of the parameter space. This targeted approach significantly reduces the number of trials needed to find the optimal hyperparameters.

Optuna also supports various other sampling methods, including grid and random search, making it highly versatile. Additionally, it provides a user-friendly interface and visualizations to track the optimization process, enabling practitioners to gain valuable insights into hyperparameter tuning. With its powerful and flexible features, Optuna has become an essential tool in machine learning projects [108].

4 METHOD

The following sections provide a detailed exploration of the methodologies employed in this research. The discussion begins with the data pre-processing techniques used to prepare the raw data for accurate and effective modelling, including strategies for addressing challenges such as missing values and normalization. The chapter then outlines the approach taken to develop and evaluate baseline models, followed by a comprehensive explanation of the ST-GCN architecture. Finally, the transfer learning strategies, including the implementation of the BAM adapter, are discussed, highlighting their role in enhancing model performance across different charging station networks.

4.1 PRE-PROCESSING

Due to its intermittent and irregular nature, the ChargeFinder dataset requires applying several pre-processing techniques to be suitable for time series forecasting. The primary challenge lies in transforming the data to address the intermittent nature that comes from the logging of user queries. To effectively handle these irregular intervals, various data imputation techniques are applied to resample the dataset into a continuous form. Moreover, data cleaning is also important to enable the best performance possible from the forecasting methods. This involves handling missing values and correcting faulty values. This pre-processing stage is critical for answering the first research question, which focuses on identifying the most effective data cleaning, transformation, and imputation methods to manage the dataset's inherent intermittency and enhance the reliability of occupancy forecasting.

When analyzing the number of chargers at each station over time, it becomes evident that this quantity fluctuates, as shown in Figure 5 on page 12. To ensure consistency in the analysis, the occupancy data is normalized by recalculating occupancy as `occupied_count` divided by `outlet_count`. This normalization accounts for variations in the number of outlets available at each station during different time periods, providing a more accurate representation of station occupancy levels for comparative analysis.

This research project has experimented with three different data configurations to identify the most effective approach for forecasting using the ChargeFinder data. Exploring these configurations aims to determine the optimal method for accurately predicting EV charging station occupancy and ensuring the robustness of the forecasting model.

- **Data with imputations**: The first data configuration is the most straightforward to implement, involving the resampling of intermittent time series data into a continuous time series using the data imputation methods discussed in Section 3.1. The resampled dataset is a univariate time series that only contains occupancy for a station. The model forecasts the charging station occupancy only by using historical charging station occupancy data. The imputation methods used for resampling the ChargeFinder dataset are discussed in Section 3.1 and they are LOCF, NOCB, and k-NN. According to [110], a jagged time series is more challenging to predict than a smooth one due to its higher variability and fewer consistent patterns. Smooth time series are generally more predictable because they allow for more accurate identification of trends and patterns. In contrast, jagged time series, characterized by high variability and irregularities, tend to obscure trends and increase noise, making predictions more difficult. All imputation methods modify the data to create a continuous time series, and the most suitable method is the one that allows the forecasting model to produce the most accurate predictions while still closely resembling the original data. Each imputation method will be tested to reconstruct a continuous dataset from the irregularly spaced data points, allowing for reliable time series forecasting.

- **Auxiliary data with imputations**: The data from ChargeFinder includes the seven features listed in Table 2. However, not all of these features are suitable for forecasting occupancy. For instance, `available_count` and `occupied_count` are inversely related, making the inclusion of both in the forecasting model redundant. Additionally, the `created_at` feature is discarded during the resampling process, and `unknown_count` does not relate to occupancy. To enhance the accuracy of occupancy forecasting, the thesis also attempted to incorporate auxiliary data into the analysis (see more details in Section 4.1.2). However, the initial results using the auxiliary data were not promising. Due to time constraints, further exploration of this approach was not pursued and would fit as a potential future work, as explained in Section 7.2.

- **Intermittent data.** This project also explores whether data without imputation is better suited for forecasting. Since data points are logged when users query the ChargeFinder website and app, the density of data points could provide useful information about the arrival of EV drivers at charging stations. However, due to the intermittent nature of the data, most traditional models, such as LSTM or machine learning methods, require imputation to handle gaps in the data. For the Transformer model in this project, one alternative to make it able to handle intermittent data, allowing it to handle gaps without imputation, is to use *fractional positional encoding*. More details on fractional positional encoding are provided in Section 3.3.2.1.

In the intermittent data configuration, the input to the Transformer consists of intermittent charging station data that has not been resampled or processed with any data imputation techniques. The positions for the output sequence can be set according to any desired intervals. Therefore, it was decided that the output of the Transformer should be a continuous time series, consistent with the output generated when using continuous input data. This continuity is achieved by encoding the target sequence generated by the intermittent Transformer with positions spaced at 5-minute intervals. By ensuring that both the intermittent Transformer and the regular Transformer produce continuous time series outputs, the comparison of results becomes more straightforward and meaningful.

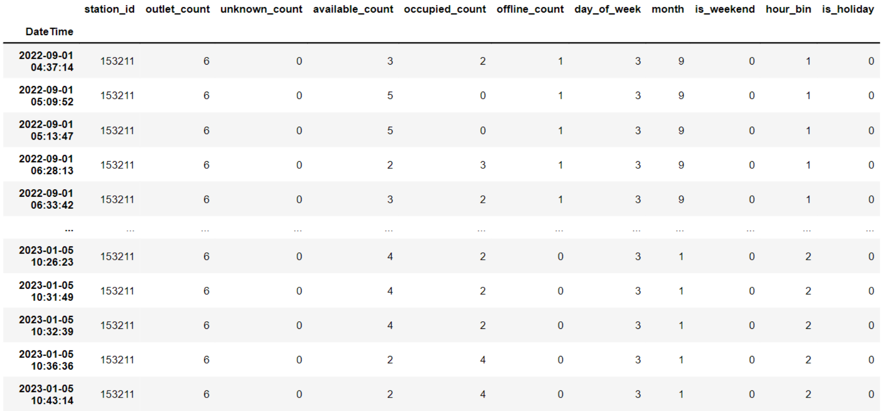
After addressing the missing and faulty data issues detailed in Section 2, each data configuration will be tested using various machine learning and deep learning methods. The data configuration that achieves the best performance in these tests will be chosen for the final implementation of the ST-GCN model.

4.1.1 *Stations*

As outlined in Table 1, this research project accessed data from a total of twenty-nine stations. However, there is a significant variance in the number of data points attributed to each station. For instance, the station Hälsingland_19 in Malmö has only 439 data points, while IONITY in Värnamo has $71,024$ data points. When applying data imputation to stations with a low number of data points, trends may become less distinguishable or even disappear. Consequently, it was decided to include only the five stations with the most data points per location in this project.

4.1.2 *Auxiliary Data*

When forecasting charging station occupancy with the data collected from ChargeFinder with a multivariate model, five features can be used: `outlet_count`, `unknown_count`, `available_count`, `occupied_count`, and `offline_count`. To supplement this, additional features are added to the data to enhance the analysis: `day_of_week` and `month` are included to capture temporal patterns over weeks and months. Addi-



	station_id	outlet_count	unknown_count	available_count	occupied_count	offline_count	day_of_week	month	is_weekend	hour_bin	is_holiday
DateTime											
2022-09-01 04:37:14	153211	6	0	3	2	1	3	9	0	1	0
2022-09-01 05:09:52	153211	6	0	5	0	1	3	9	0	1	0
2022-09-01 05:13:47	153211	6	0	5	0	1	3	9	0	1	0
2022-09-01 06:28:13	153211	6	0	2	3	1	3	9	0	1	0
2022-09-01 06:33:42	153211	6	0	3	2	1	3	9	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
2023-01-05 10:26:23	153211	6	0	4	2	0	3	1	0	2	0
2023-01-05 10:31:49	153211	6	0	4	2	0	3	1	0	2	0
2023-01-05 10:32:39	153211	6	0	4	2	0	3	1	0	2	0
2023-01-05 10:36:36	153211	6	0	2	4	0	3	1	0	2	0
2023-01-05 10:43:14	153211	6	0	2	4	0	3	1	0	2	0

Figure 6: An extract of the processed dataset.

tionally, `is_weekend` indicates whether a day is part of the weekend, `hour_bin` categorizes the time of day into hour intervals to analyze daily usage patterns, and `is_holiday` identifies public holidays. The `is_holiday` feature is created by considering public holidays in Sweden throughout the period considered for the analysis [111]. This includes the period between December 25th and January 1st, recognizing that many people are on holiday during this time. Figure 6 depicts an illustrative extract of the processed dataset. Additionally, although weather data from the Swedish Meteorological and Hydrological Institute (SMHI) [112], containing various weather features, was considered for incorporation into the dataset, it was ultimately not included in the final analysis due to time constraints and the fact that adding more features did not initially lead to any improvement, leaving this as a potential future work. A correlation matrix is generated to determine if there are any significant correlations between the weather features and the occupancy rates of EV charging stations. For instance, certain weather features, like temperature, might exhibit strong correlations with fluctuations in occupancy, as colder conditions can lead to higher demand due to faster battery drainage. Features with high correlation coefficients are retained for further modelling, as they are likely to provide valuable information for predicting occupancy. Conversely, features with low correlation coefficients are discarded.

4.1.3 *Graph*

As part of the pre-processing, a graph was constructed for the ST-GCN to process the input data. The graph structure was designed as a simple fully connected network, rather than a graph that fully models the road network connecting the charging stations. This decision was made to avoid the additional complexity that would arise when fine-tuning a model from one charging network to another. In the context of transfer learning, the weights of a GNN are fine-tuned for the new

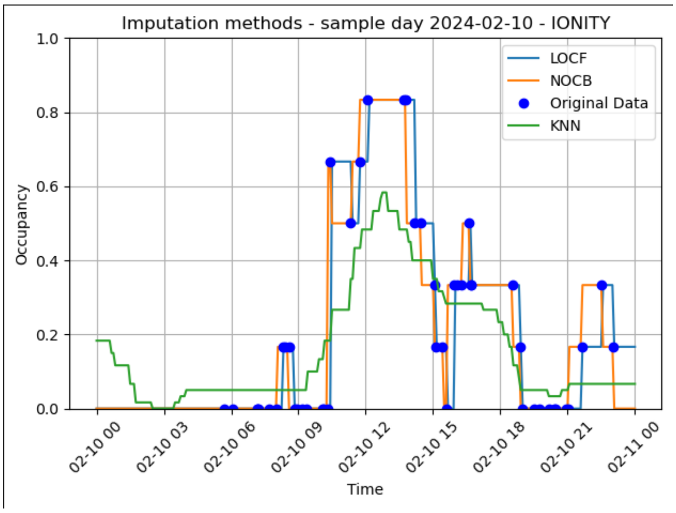


Figure 7: An overview of the diverse imputation methods on a sample day, 2024-02-10 on the IONITY dataset.

charging network. However, if the graph structure were to change, the weights could not be reused effectively.

The graph used for the charging station networks also includes the distances between the stations. These distances were collected using Google Maps by selecting two stations and recording the shortest possible driving distance between them [113].

4.2 FORECASTING ML AND LINEAR METHODS

ML techniques are widely utilized for forecasting due to their simplicity and effectiveness in capturing patterns within data. The following subsections present and explain the ML models employed in this project as baseline methods.

An autocorrelation plot, as shown in Figure 9, is generated to visually assess the data and guide the selection of initial parameters for the machine learning models employed in this project. These insights help ensure that the ML techniques are effectively tailored to the data.

In comparing traditional ML methods such as MA, AR, and ARIMA with DL models, a similar strategy to the one used for the DL models, described in Section 4.3, is adopted. Initially, the MA, AR, and ARIMA models will be fitted to the first n data points (from t to $t-n$) to predict the subsequent L data points (from t to $t+L$). Subsequently, the fitting data and prediction window will be updated using the same sliding window technique as for the DL methods, followed by model retraining. Finally, the average MSE value across all sequences will be computed for each model, serving as a comprehensive evaluation metric for comparison. This iterative process will ensure a fair comparison throughout all the ML and DL baseline models.

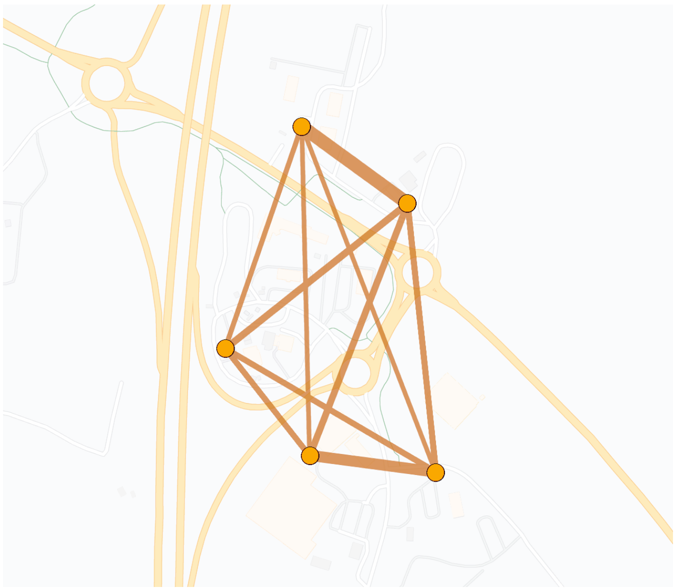


Figure 8: An image of the Värnamo charging network where the graph is overlayed on the Google Maps website. (Note: The thickness of an edge is proportional to its weight.)

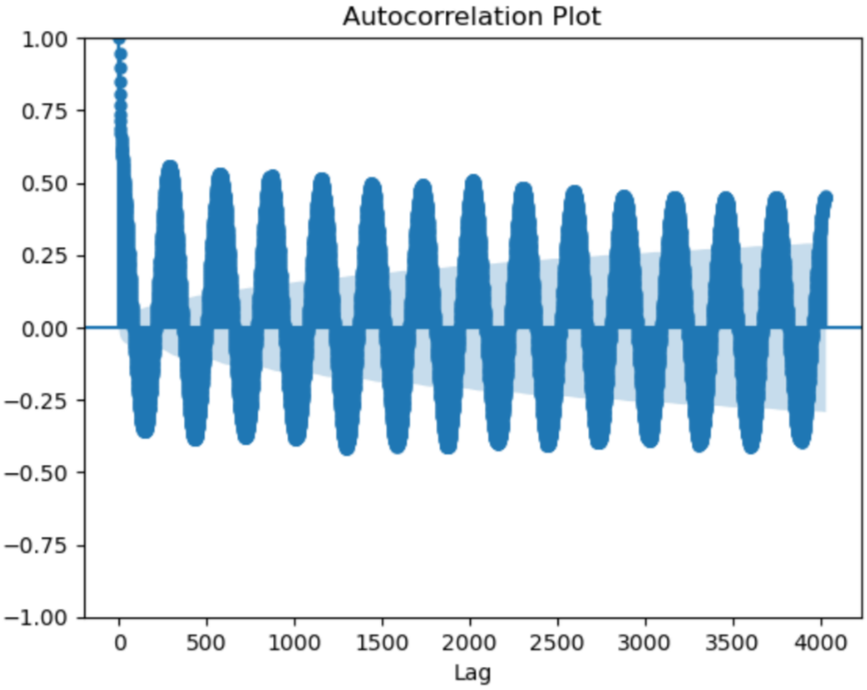


Figure 9: Autocorrelation plot of the data from the test station.

4.2.1 *Horizontal line*

The *horizontal line* forecast, also known as the *persistence forecast* or *Last Value Carried Forward (LVCF)*, is a straightforward yet effective baseline method in time series forecasting. It predicts that future values of a time series will remain unchanged, equal to the most recent observation. This approach essentially extends a horizontal line from the last known data point, assuming stability in the immediate future.

The primary strength of the horizontal line forecast lies in its simplicity and minimal assumptions. It is easy to implement, requires no complex computations, and serves as a valuable benchmark against which more advanced models can be evaluated. If a more sophisticated model fails to outperform this baseline, it may indicate that the added complexity is unwarranted.

4.2.2 *MA*

MA models aim to understand how the current value of a time series relates to past errors in prediction. An MA model of order q (i.e., $MA(q)$) predicts the current value by considering a weighted average of recent prediction errors up to a certain number of time steps (specified by q) [29], as shown in Equation 3:

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i}, \tag{3}$$

where x_t represents the value of the time series at time t, μ is the constant term, $\theta_0, \theta_1, \ldots, \theta_q$ are the coefficients associated with the lagged error terms up to lag q, and ε_{t-i} represents the lagged error terms at time $t-i$.

4.2.3 *AR*

AR models focus on modeling the relationship between the current value of a time series and its past values. AR models of order p (i.e., $AR(p)$) predict the current value based on a linear combination of the p most recent past values [29], as shown in Equation 4:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t, \tag{4}$$

where x_t represents the value of the time series at time t, c is the constant term, $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients, x_{t-i} denotes the lagged values of the time series up to lag p, and ε_t represents the error term at time t.

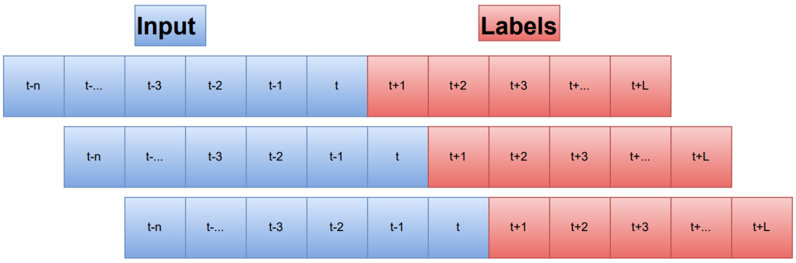


Figure 10: An illustration of three sequences resulting from the application of the sliding window technique twice. t = present time, n = input sequence size, L = output sequence size

4.2.4 *ARIMA*

ARIMA, denoted as $ARIMA(p, d, q)$, combines AR, differencing (I), and MA components to model time series data, as shown in Equation 5.

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + t + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i}. \tag{5}$$

Here x_t represents the value of the time series at time t, c is the constant term, $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients, x_{t-i} denotes the lagged values of the time series up to lag p, t represents a trend term. $\theta_0, \theta_1, \ldots, \theta_q$ are the coefficients associated with the trend term lagged up to lag q, and ε_{t-i} represents the error term at time $t - i$.

4.3 FORECASTING DL METHODS

In the forthcoming subsections, the DL methodologies implemented in this project are presented. It's important to note that the input size of the sequences for these models will also be determined based on insights from the same autocorrelation plot referenced in Section 4.2 (see Figure 9).

The DL models will be trained on data loaders, consisting of sequential inputs as depicted in Figure 10, where 3 sequences are shown as an example. Each sequence will comprise an input sequence with n datapoints containing all the features, and an output sequence containing labels representing the subsequent L occupancy datapoints, equivalent to the upcoming 3 hours. To ensure comprehensive coverage of the dataset, a sliding window technique with a step size of 1 datapoint will be employed to extract all possible sequences.

4.3.1 *LSTM*

The LSTM model in this project is implemented as a multi-step predictor, capable of forecasting future values based on historical input sequences. It comprises multiple layers of LSTM cells followed by a fully connected layer to map the LSTM outputs to the desired prediction space. The LSTM layer will iterate through the input sequence, and then continue generating predictions based on its own previous predictions and cell states. The resulting LSTM resembles a many-to-many architecture configuration, as depicted in Figure 11.

The LSTM model incorporates several components, including input gates (i_t), forget gates (f_t), output gates (o_t), and memory cells (C_t). These components work together to process sequential data and make predictions for the next time step. The equations describing the entire process are presented in the following (from Equation 6 to 11), as reported in [114].

$$f_t = \sigma(X_t U^f + S_{t-1} W^f + b_f) \tag{6}$$

$$i_t = \sigma(X_t U^i + S_{t-1} W^i + b_i) \tag{7}$$

$$\tilde{C}_t = \tanh(X_t U^c + S_{t-1} W^c + b_c) \tag{8}$$

$$C_t = C_{t-1} \odot f_t + i_t \odot \tilde{C}_t \tag{9}$$

$$o_t = \sigma(X_t U^o + S_{t-1} W^o + b_o) \tag{10}$$

$$S_t = o_t \odot \tanh(C_t) \tag{11}$$

Here, Equation 6 is the forget gate that controls the information flow from the previous cell state C_{t-1}; Equation 7 is the input gate that controls the information flow to the current cell state C_t; Equation 8 is the candidate cell state that contains new candidate values; Equation 9 is the updated cell state that combines the previous cell state with the candidate values; Equation 10 is the output gate that controls the information flow from the current cell state C_t, and Equation 11 is the hidden state output that contains the filtered information based on the output gate.

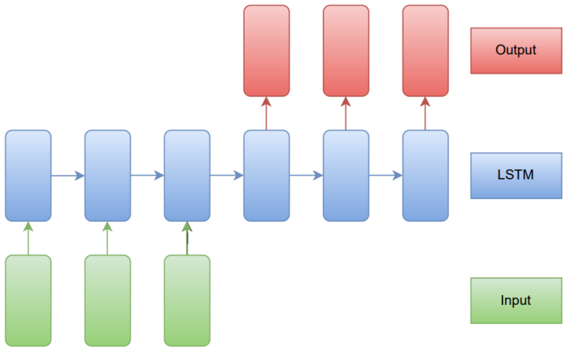


Figure 11: Many-to-many LSTM architecture configuration.

4.3.2 *Transformer*

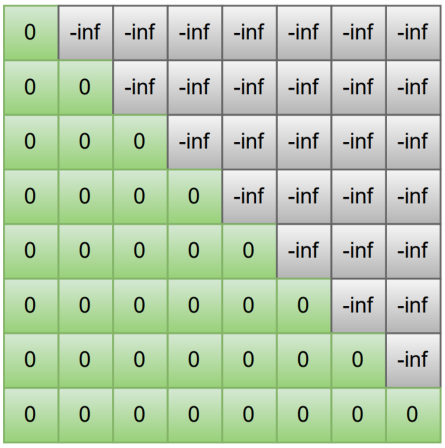
This project leverages the Transformer architecture as detailed in the seminal work by Vaswani et al. [68]. The choice of this architecture is because of the underlying similarities between language processing and the dynamics of time-series data. Both fields use sequences of data where the overall context significantly influences the importance of each element. In language, the value of a word can shift based on its position or relation to other words. Similarly, in time-series data, the significance of a data point is shaped by its predecessors and successors, underscoring temporal patterns.

The embedding layer from the original Transformer model has been omitted in this work since the inputs for this project are inherently numerical, the transformation typically provided by the embedding layer is unnecessary. This removes a redundant processing step, thereby allowing direct engagement with the numerical input data.

To ensure the input sequence includes temporal information, positional encoding is employed since Transformers process inputs in parallel and lack inherent sequence awareness. In this thesis, except for when testing to forecast with an intermittent dataset, the general positional encoding proposed in the original Transformer paper [68] is implemented, which incorporates sinusoidal and cosinusoidal variables into the input. This strategy enables the model to discern the sequence of events, akin to recognizing the rhythm in a piece of music. Please see Equation 12 and 13, where d_{model} is the dimensionality of the model's input.

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{12}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right). \tag{13}$$



0	-inf	-inf	-inf	-inf	-inf	-inf	-inf
0	0	-inf	-inf	-inf	-inf	-inf	-inf
0	0	0	-inf	-inf	-inf	-inf	-inf
0	0	0	0	-inf	-inf	-inf	-inf
0	0	0	0	0	-inf	-inf	-inf
0	0	0	0	0	0	-inf	-inf
0	0	0	0	0	0	0	-inf
0	0	0	0	0	0	0	0

Figure 12: Visualization of the causal mask implementation.

The multi-head attention mechanism, explained in Section 3.3.2, is described using formulas 14 and 15:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V, \tag{14}$$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \dots, \mathrm{head}_h)W^O. \tag{15}$$

Here Q represents the matrix of queries, K is for the matrix of keys, and V represents the matrix of values. In addition, QK^T is the dot product of the query matrix with the transpose of the key matrix, and $\sqrt{d_k}$ is the square root of the dimensionality of the keys. head_i represents the output of the i-th attention mechanism, W^O is the output weight matrix, and h is the number of heads in the multi-head attention mechanism.

Furthermore, the masking strategy is tailored to preserve the model's predictive accuracy. By shifting the decoder's input one timestep to the right and applying a causal mask, the model is constrained to use only preceding information for making predictions. The causal mask, detailed mathematically in Equation 16, and visually in Figure 12, acts as a barrier against future data, ensuring predictions are grounded in past and present context. As a result, this causal masking is applied to the input sequence of the decoder. [76][115]

$$\mathrm{Mask}(i, j) = \begin{cases} 0 & \text{if } i \leqslant j, \\ -\infty & \text{otherwise.} \end{cases} \tag{16}$$

The head of the Transformer needs to be altered since the model is designed for a regression-based approach, generating numerical values directly. This is often done by removing the softmax layer [49][116], usually indicative of classification tasks, and retaining only

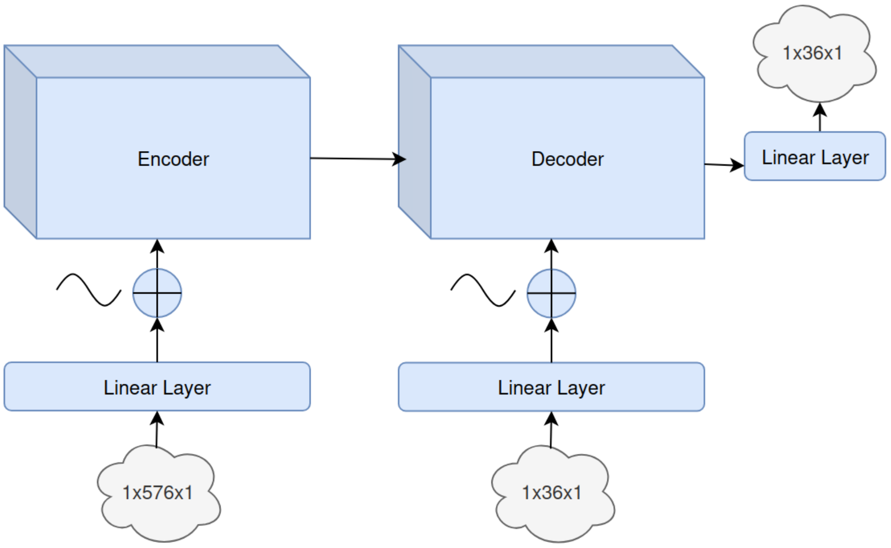


Figure 13: Visualization encoder-decoder structure utilized in this thesis. The cloud shapes represent the tensors' dimensions of the input and output tensors. The first number indicates the number of stations, the second represents the length of the sequence, and the third denotes the number of features.

the linear layer as output. This modification aligns with the project's aim to produce continuous variable predictions rather than discrete categorizations. This is described in Equation 17, where H represents the final hidden states from the Transformer's decoder, W_o is the weight matrix of the output linear layer, and b_o is the bias.

$$\text{Output} = W_o H + b_o \tag{17}$$

Attention, as outlined in Equation 14, is a fundamental component of both the encoder and decoder in the Transformer model. The encoder employs self-attention to process the input sequence, generating an encoding that captures the contextual relationships between the timesteps. The decoder, on the other hand, uses self-attention to process its input—the shifted target sequence—and cross-attention to integrate the encoder's output. This mechanism effectively connects the encoder and decoder, allowing the model to leverage contextual information from both the input and target sequences.

Figure 13 shows an overview of the Encoder-Decoder model with the aforementioned changes implemented.

During the training phase of Transformer models, a particular learning rate scheduler is proposed in the seminal paper [68]. This scheduler, detailed in Equation 18, is designed to optimize the training process by adjusting the learning rate according to the number of training steps completed. The formula specifies a dynamic learning rate adjustment mechanism where the rate initially increases linearly during the first set of warmup steps. Subsequently, the learning rate decreases proportionally to the inverse square root of the step number. This approach is described in the following equation:

$$\text{lrate} = d_{\text{model}}^{-0.5} \cdot \min\left(\text{step\_num}^{-0.5}, \text{step\_num} \cdot \text{warmup\_steps}^{-1.5}\right). \tag{18}$$

Here, $lrate$ represents the learning rate, d_{model} denotes the model's dimensionality, $step_{\text{num}}$ is the current step number, and $warmup_{\text{steps}}$ refers to the predefined number of steps during which the learning rate is linearly increased. This learning rate scheduling method is specifically crafted to enhance the model's learning efficiency by facilitating a smoother and more adaptive adjustment of the learning rate throughout the training process. It embodies a careful balance between accelerating the initial learning pace and ensuring a gradual reduction in the learning rate to stabilize the learning process as training progresses [68].

4.3.3 *Intermittent Transformer*

One of the analyses in this thesis involves using the raw intermittent data provided by ChargeFinder without applying any data imputation techniques. To effectively handle this data within the Transformer architecture, a positional encoding method capable of encoding non-discrete positions is required. This research employs the *fractional positional encoding* method introduced in [78]. Fractional positional encoding is similar to the positional encoding introduced in [68], but instead of restricting the data point to discrete positions along a sine wave, it allows the data point value to be positioned anywhere along the sine wave. More details on this method can be found in Section 3.3.2.1.

The generation of input and output sequences was modified to handle intermittent data and output corresponding positions when using this data in the Transformer. Typically, when a sliding window is used to convert the dataset into sequences for training, as shown in Figure 10, the position within the dataset is indexed by an integer. However, when generating intermittent sequences, DateTime objects are used for indexing instead. Given a DateTime index, the subsequent 3-hour intervals are then generated from the dataset that is resampled using k-NN. As for the input data, two methods were tested:

- **Fixed Length Window**: The input intermittent sequence is generated by selecting all the data points within a 2-day window preceding the given DateTime index. However, this also leads to a class imbalance due to there being a varying amount of data points in each 2-day window. This if fixed by subsampling the amount of data points in the sequence that have higher amounts of data points.

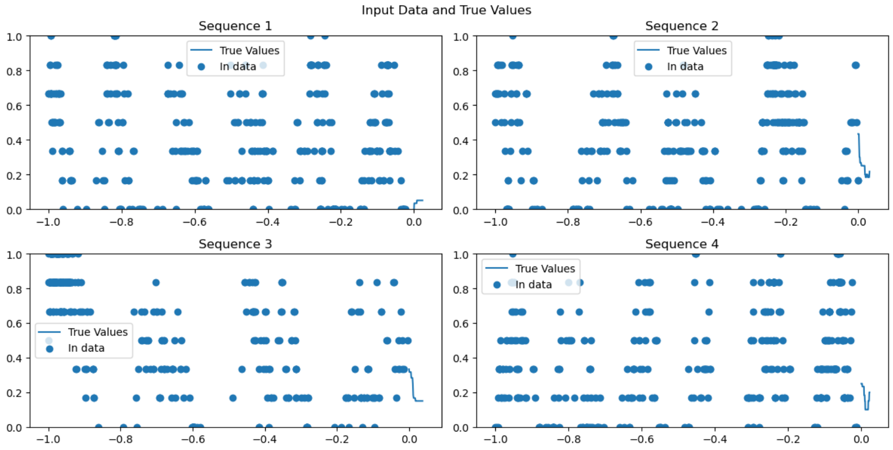


Figure 14: The intermittent input and continuous output used by the intermittent Transformer. These sequences are generated using data from the IONITY charging station.

- **Fixed Amount of Data Points**: The input intermittent sequence is generated by selecting a fixed number of preceding data points. This fixed number is determined based on the average number of data points required to fill a 2-day window, which is approximately 360 data points. However, due to the intermittent nature of the dataset, the timespan of the input sequence will vary in length. To address this, fractional positional encoding is used. By scaling the positions of the input and output sequence, it is ensured that a 5-minute difference between the positions are the same in the positions for the input data and the output data.

The resulting input and output sequence is shown in the plot in Figure 14.

4.3.4 *GCN*

A GCN operates on graph-structured data $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of N nodes and $\mathcal{E}$ is a set of edges. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents the connections between nodes, and $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the feature matrix, where F is the number of features per node. Equation 19 shows the forward propagation where $\mathbf{H}^{(l)}$ is the input feature matrix at layer l, $\mathbf{H}^{(0)}$ is the input to the neural network, $\hat{\mathbf{A}}$ is the normalized adjacency matrix, $\mathbf{W}^{(l)}$ is a trainable weight matrix for layer l, and σ is an activation function such as ReLU.

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{19}$$

When implementing GCNs in PyTorch, the simplest method is to use the *PyTorch-Geometric* library [117], which contains the *GCNConv* layer. With this library, a layer can be declared by setting the number of input and output features. It can then be used by passing the edge indices and edge weights, which is another way to represent the adjacency matrix as parameters, along with the input data.

4.3.5 *ST-GCN*

The ST-GCN in this project is a hybrid network that consists of a GCN and a Transformer. The input data for the GCN will be structured such that each node in the graph contains a sequence of ChargeFinder data corresponding to each station. This approach simplifies data processing compared to having each node containing data for one timestamp datapoint and results in multiple graphs, due to the design of the torch_geometric API [117].

The Transformer leverages an Encoder-Decoder architecture, as explained in Section 4.3.2. Instead of directly inputting the charging station occupancy data over time, the output from the GCN is used as input to the Transformer. The Transformer then predicts a sequence for each node individually. This approach is used to optimize GPU memory usage, as opposed to using five separate Transformers for each node or one large Transformer that outputs a sequence where each output corresponds to the occupancy of a different charging station. The Transformer used in the ST-GCN is essentially the same as the one used outside the ST-GCN, with the primary difference being that it has a higher input feature size to account for the time series data enriched with spatial information from the GCN. However, the Transformer requires the last token in the input sequence to generate an output. This poses a challenge because the input data to the Transformer has been transformed into a different feature space by the GCN, meaning it differs from the occupancy values that the Transformer predicts. As a result, the Transformer cannot use the last value of its input directly to generate the output sequence. To address this issue, the ST-GCN passes the last value of the input sequence from the GCN as a separate variable to the Transformer. The Transformer then uses both the last value of the GCN input and the GCN output to generate the occupancy prediction. The final ST-GCN architecture will resemble the network depicted in Figure 15.

In summary, the two components of the proposed method are as follows:

- GCN:
  - The spatial processing module of the ST-GCN is comprised of multiple GCN layers. These layers operate on the underlying graph structure, capturing spatial dependencies among nodes in the graph.
  - Each GCN layer applies graph convolution operations followed by non-linear activation functions and dropout reg-

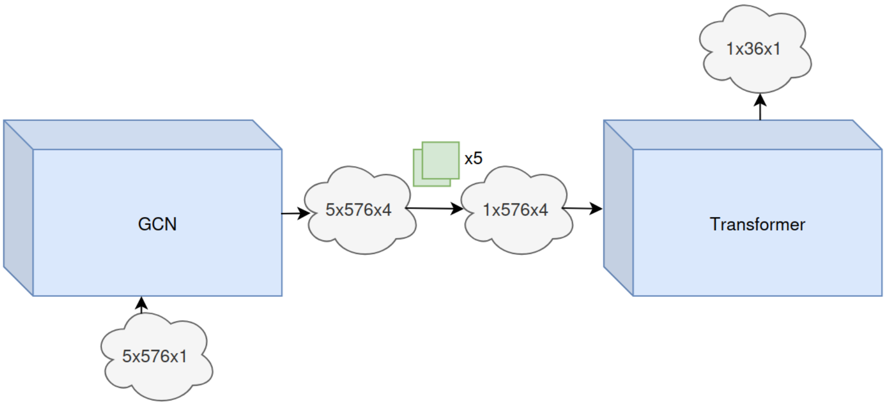


Figure 15: Visualization of the ST-GCN architecture implemented in this thesis. The cloud shapes represent the tensors' dimensions at different stages within the ST-GCN. The first number indicates the number of stations, the second represents the length of the sequence, and the third denotes the number of features. The Transformer generates predictions sequentially, one by one, for each charging station.

ularization to extract hierarchical features from the input node features.

- Transformer:
  - For temporal processing, ST-GCN utilizes a Transformer architecture, which employs sequence-to-sequence generation to forecast occupancy based on the temporal aspect of the data.
  - The Transformer processes the time series for each charging station in the network one-by-one.

4.4 TRANSFER LEARNING

This section details the implementation of various fine-tuning methods, which are compared in Section 5. The methods explored include: using a pre-trained model without any fine-tuning, training the model from scratch, fully fine-tuning all parameters, fine-tuning only the last GCN layer, and fine-tuning only the last Transformer layer. Additional information on these original methods is provided in Section 4.4.1.

Additionally, various adapter methods are implemented and compared, including the MLP adapter, Tiny-Attention Adapter, and a custom adapter developed for this research project, named BAM. The MLP and Tiny-Attention Adapters serve as baselines, alongside the three common fine-tuning methods mentioned earlier. In all the fine-tuning techniques, layers that are not being fine-tuned are frozen by setting *param.requires_grad = False* in PyTorch, ensuring that their parameters remain unchanged during training. This approach allows the model to adapt to new data while preserving the already learned parameters. Detailed descriptions of the MLP adapter, Tiny-Attention Adapter, and BAM adapter can be found in Sections 4.4.2, 4.4.3, and 4.4.4, respectively.

4.4.1 *Original Fine-tuning*

Full Fine-Tuning ($\mathcal{M}_A \xrightarrow{\text{Fine-tune all}} \mathcal{M}_B$) involves adjusting all parameters of the model. While this approach can yield strong results, it is computationally expensive due to the large number of parameters being updated for each downstream task.

To mitigate this, the second approach focuses on fine-tuning only the last layer of the model, applied in two scenarios: fine-tuning the last GCN layer ($\mathcal{M}_A \xrightarrow{\text{Last GCN}} \mathcal{M}_B$) and fine-tuning the last Transformer layer ($\mathcal{M}_A \xrightarrow{\text{Last Transformer}} \mathcal{M}_B$). Fine-tuning the last GCN layer adjusts the weights of low-level data-specific features learned by the GCN, enhancing its performance on the new dataset while retaining high-level features from a broader dataset. Similarly, fine-tuning the last Transformer layer adjusts the low-level temporal features rather than the spatial features.

In addition to these three fine-tuning methods, two other methods serve as baselines: testing a model pre-trained on Network A directly on Network B without any fine-tuning ($\mathcal{M}_A \rightarrow \mathcal{M}_B$) and training from scratch on Network B ($\mathcal{M}_B \rightarrow \mathcal{M}_B$).

4.4.2 *MLP Adapter*

While fine-tuning only the last layer can reduce the number of trainable parameters, it may not yield optimal results, and the reduction in trainable parameters might not be significant enough. To address this, adapters have been introduced to further minimize the number of trained parameters while optimizing transfer learning performance. One such example is the standard "MLP adapter," originally proposed in [95]. A visualization of this adapter can be found in Figure 16. It consists of two main components: a linear reduction layer that compresses features into a more efficient bottleneck space, and an expansion layer that upsamples the data back to its original size. After the reduction, the features pass through a ReLU activation function, introducing non-linearity. Additionally, the adapter includes a residual connection that adds the original input to the upsampled output, ensuring consistency with the pre-trained model's output.

The intuition behind this design is its ability to balance compression, non-linear transformation, and information preservation. The

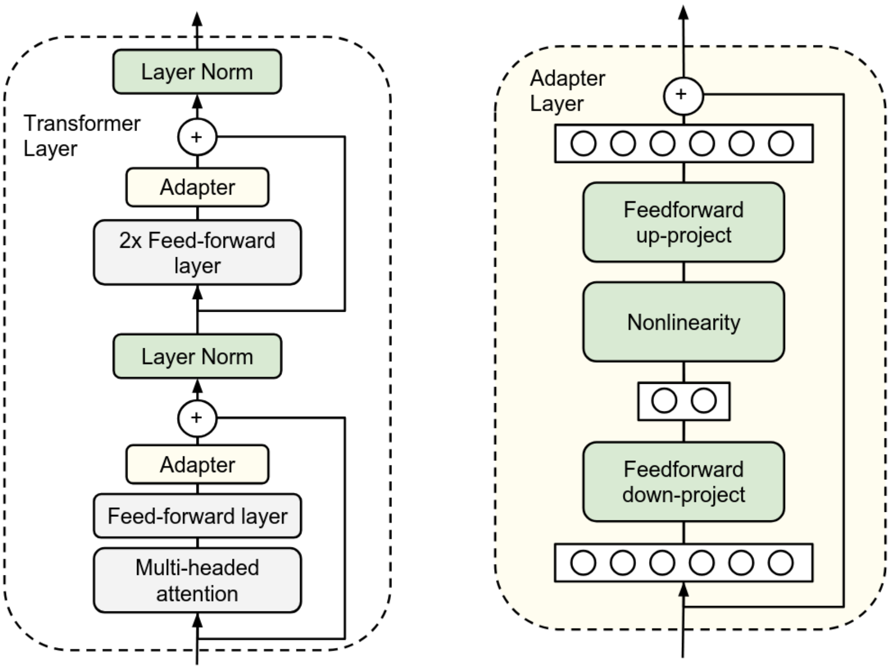


Figure 16: Architecture of the MLP Adapter, adapted from [95]. Grey and green layers are frozen, while yellow layers are trainable. The figure on the left shows the integration of the adapter into the Transformer architecture, while the figure on the right provides a detailed view of the adapter.

linear reduction layer forces the model to focus on the most critical aspects of the data by reducing dimensionality, simplifying the problem and potentially reducing overfitting. The ReLU activation adds non-linearity, enabling the model to capture more complex patterns. The expansion layer then restores the features to their original size, ensuring compatibility with the pre-trained model. Finally, the residual connection preserves the integrity of the original input, stabilizing the training process and maintaining good performance without deviating too far from the pre-trained model's original behaviour.

The adapter is integrated within the Transformer encoder layers, with two adapters per layer, placed after each feedforward network. This helps transform the feedforward network outputs into a suitable representation for the new task. However, placing the adapter before normalization might disrupt the consistency of learned scales, as the normalization layer adjusts the feedforward network's output. This placement can lead to less stable and predictable modifications, potentially reducing the effectiveness of fine-tuning by missing the benefits of the pre-trained normalization layer. Additionally, it can destabilize training because the normalization's ability to maintain consistent gradients is compromised, making training less efficient by applying transformations before achieving gradient stability. Furthermore, since the adapter consists only of feedforward networks,

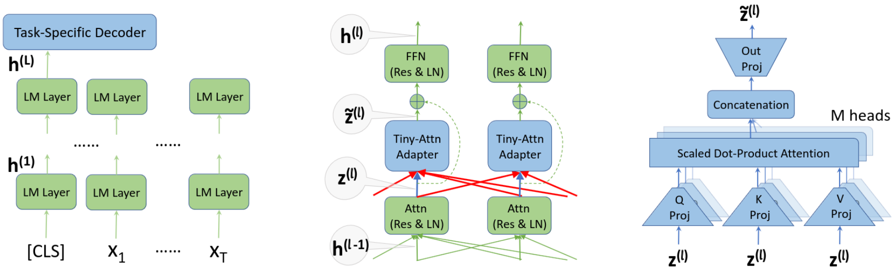


Figure 17: Architecture of the Tiny-Attention Adapter, adapted from [96]. Green layers are frozen, while blue layers are trainable.

it lacks attention mechanisms and does not capture global context information about the data points.

4.4.3 *Tiny-Attention Adapter*

To address the lack of global context in traditional adapters, [96] introduced the Tiny-Attention Adapter, visualized in Figure 17. This adapter incorporates an attention mechanism to provide global context to the input.

The architecture of the Tiny-Attention Adapter is similar to the MLP adapter described in the previous section but includes additional attention mechanisms. Rather than merely downscaling, the adapter first generates key, query, and value vectors, then downscales them, computes attention scores using multiple heads as shown in Equations 14 and 15, and finally performs an up-scaling operation. The bottleneck size is set to one, as the authors argue that a smaller bottleneck feature size results in fewer parameters. They also suggest that smaller models can perform as well as larger ones if they effectively leverage a broader context and that feed-forward adapters can achieve competitive results using minimal parameter matrices.

The Tiny-Attention Adapter is positioned after the self-attention is computed in the encoder layers and after the residual and layer normalization layers. This placement addresses the potential issue of placing the adapter before a pre-trained layer normalization, ensuring consistent performance. Additionally, this adapter is not the only component being trained during fine-tuning; the authors also train a task-specific decoder alongside the adapters, resulting in much higher parameter usage than is typically desired in fine-tuning adapters.

4.4.4 *Bridged Attention Module*

To address the lack of adapters that consider both temporal and spatial information, this project introduces BAM, a custom adapter specifically designed to seamlessly integrate the GCN and Transformer components within the ST-GCN model. BAM also serves to facilitate the connections between encoder layers and subsequent layers within the Transformer architecture. The primary goal of BAM is to create a unified adapter that effectively handles the complex interactions between spatial and temporal data while ensuring a smoother flow of information between the layers in the Transformer.

In the ST-GCN model, the BAM adapter serves as a critical bridge within the Transformer, linking encoder layers and connecting the output of the last encoder layer with the decoder layers. When positioned between the GCN and Transformer, BAM includes the last layer of the GCN at its beginning and fine-tunes it to enhance performance, as shown in Figure 18. This initial fine-tuning step improves performance while introducing no new parameters and only minimally increasing the number of trainable parameters.

BAM was designed to adhere to the standard bottleneck structure for adapters while incorporating feedforward networks and attention mechanisms, similar to the Tiny-Attention adapter, to capture global context. The use of attention in BAM is crucial as it enables the model to focus on relevant temporal sequences and spatial features, ensuring that critical information is effectively communicated across the model.

When BAM is inserted as a bridge between the GCN and the Transformer, it ensures that the sequence-wise attention mechanism effectively prepares the GCN output for the Transformer's temporal processing. Additionally, BAM is placed at the top of each encoder layer to introduce sequence-wise attention, facilitating smooth transitions between encoder layers and subsequent layers, whether they are additional encoders or decoders. This strategic placement enhances the flow of crucial temporal information from the encoder to the next stages of the model.

Unlike the MLP adapter and Tiny-Attention adapter, BAM is positioned on top of the encoder layers, specifically after the pre-trained layer normalization, as shown in Figure 18. This placement ensures that the learned scales and shifts of the layer normalization weights are preserved, preventing disruptions that could occur if these learned transformations were applied to unintended data. To stabilize training, BAM includes its layer normalization at the end, and to prevent overfitting, particularly with smaller downstream datasets, it incorporates a dropout layer.

Furthermore, BAM performs unified downscaling on the input before generating the queries, keys, and values, resulting in fewer trainable parameters compared to the Tiny-Attention adapter, which applies downscaling after these elements are created. A scaling parameter is included at the end of BAM to fine-tune the output size as needed. Notably, during training, BAM is the only component being trained, unlike the Tiny-Attention adapter, which also trains a

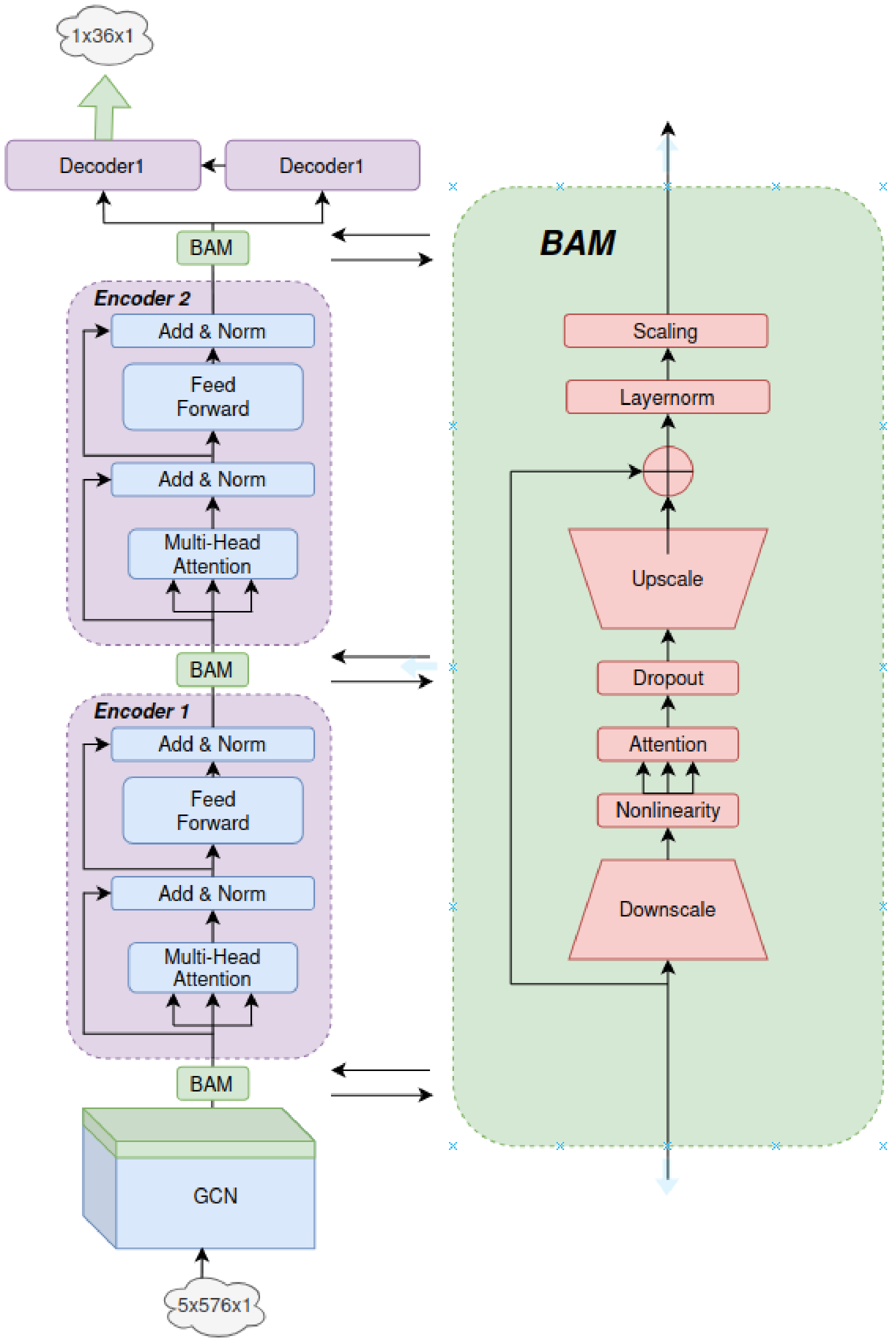


Figure 18: Architecture of BAM. On the left side, the placement of the BAM adapter within the ST-GCN architecture is presented. On the right side is the internal architecture of BAM. Blue represents frozen layers, while red and green represent trainable layers. Notice that when BAM is positioned between the GCN and Encoder 1, the last GCN layer is also fine-tuned.

task-specific decoder, thereby significantly increasing the number of trainable parameters.

Finally, BAM's architecture, with a bottleneck size of 2 instead of 1 compared to the Tiny-Attention adapter, maintains a low number of parameters while still extracting at least two important features in the bottleneck layer. Additionally, BAM includes a skip connection from the original input directly to the output after upscaling, ensuring that essential information is preserved throughout the bottleneck layer.

In summary, BAM is a strategic enhancement to the ST-GCN model, effectively bridging key components to ensure efficient processing of both spatial and temporal features. Its design minimizes computational overhead, making it valuable for fine-tuning in scenarios where limited data is available for training from scratch.

5 RESULTS

The autocorrelation plot proved essential in determining the optimal input window size for the analysis. It revealed a daily seasonal pattern, as shown in Figure 9 on page 32). This pattern indicated a strong dependency up to the previous three hours (36 data points) and a seasonal correlation with preceding days, all showing autocorrelation values around 0.5. Based on these insights, this study utilizes an input window spanning the past two days, comprising 576 data points ($n = 576$), as the historical context for forecasting 3 hours ahead. The selection of a two-day window is motivated by the need to capture daily patterns and trends, such as cyclical behaviours while avoiding the inclusion of excessive data that could lead to overfitting. This window size balances providing sufficient historical information to the model and mitigating the risk of overfitting, which may impair the model's generalization performance.

This decision was further informed by an analysis of the autocorrelation plot, which reveals a strong daily pattern in the data. The plot indicates that the most significant correlations occur within the previous two days, which fall within the highest confidence intervals. Therefore, this timeframe is optimal for capturing these recurring patterns, without overwhelming the model with unnecessary data. Conversely, using a shorter timeframe, such as the last two hours, would be insufficient to capture these critical patterns.

In alignment with the forecasting objectives, the output window was set to 36 data points, corresponding to a three-hour forecast horizon. This choice reflects the need for mid-term predictions that are both accurate and timely for decision-making processes.

5.1 PRE-PROCESSING

This section details the outcomes of the pre-processing steps applied to the data set. Each result is analyzed to understand its impact for the subsequent stages of this study.

5.1.1 *Data features*

In the initial phase of this study, additional features, such as the number of available charging ports, were considered for inclusion in the model. However, these features were redundant. The occupancy feature alone encapsulates the relationship between the number of occupied and available ports, making the inclusion of the raw count of available ports unnecessary and potentially confusing for the model.

Furthermore, weather-related features were also considered for inclusion, as they could potentially influence charging patterns. However, due to time constraints and the fact that adding more features did not initially lead to any improvement, the decision was made to defer the integration of weather features to future work.

The decision to focus solely on the occupancy feature, represented as a percentage, was made for several reasons. Primarily, using a percentage-based measure allows for normalization across charging stations of different sizes. Charging stations can vary significantly in the number of available ports, and using a raw count of occupied ports would introduce a bias toward larger stations. For example, a station with 10 charging ports where 5 are occupied would show 50% occupancy, whereas a smaller station with 5 ports and 3 occupied would show 60%. Representing occupancy as a percentage enables the model to make meaningful comparisons across stations, regardless of their absolute size. Furthermore, the percentage-based feature is more intuitive and interpretable, providing a clear, normalized indicator of station utilization without the need for additional adjustments or scaling.

5.1.2 *Forecasting horizon*

The choice of a 3-hour time horizon for predicting EV charging station occupancy is driven by a careful consideration of the trade-offs between prediction accuracy and practical utility. This time frame strikes a balance between the increasing difficulty associated with long-term forecasts and the actionable relevance of the predictions. Moreover, this decision was directed from an analysis of the autocorrelation plot in Figure 9 which revealed significant correlations within this time range, supporting the choice of a 3-hour horizon as an optimal point for maintaining both predictive accuracy and practical usefulness.

As the prediction horizon extends further into the future, the complexity and uncertainty of the forecasting task increase. Long-term predictions are inherently more challenging due to the accumulation of uncertainties over time, which can lead to reduced accuracy. This is especially true in dynamic environments like EV charging stations, where occupancy patterns can be influenced by various factors, including time of day, day of the week, and special events. As a result, the longer the forecasting horizon, the more difficult it becomes to account for all potential influences and variations.

For EV charging stations, mid-term forecasts tend to offer greater practical value and relevance compared to immediate, short-term, or long-term predictions. A 3-hour horizon provides a sufficient time

Imputation Methods	
Method	**$\overline{\text{MSE}}$**
Intermittent Transformer (Fixed Datapoints)	304.44 ± 13.02
Intermittent Transformer (Fixed Timeframe)	240.08 ± 3.29
LOCF	225.72 ± 6.38
NOCB	207.05 ± 7.10
k-NN	168.61 ± 6.15

Table 4: Comparison of the average MSE values and standard deviations obtained from training the Transformer model on IONITY data, which is resampled using the imputation techniques intermittent with fixed datapoints, intermittent with fixed timeframe, LOCF, NOCB, and k-NN All values are multiplied by 10^4.

frame to offer useful predictions while still being manageable in terms of prediction complexity. On the other hand, if a drive is short, there is often no need for a charge within this period, making the 3-hour forecast particularly reasonable given typical driving patterns and charging needs. Indeed, electric vehicles typically need to charge after around 3 hours of driving. Many modern EVs have ranges of $200 - 300$ miles on a full charge, depending on the model. For instance, a Tesla Model 3 has a real-world highway range of around 310 miles, while other vehicles like the Porsche Taycan achieve about 300 miles under similar conditions [118]. At highway speeds (around 100-110km/h), drivers would reach this range limit after approximately $3 - 4$ hours of continuous driving, aligning with typical charging needs. Thus, forecasting EV charging station occupancy over a three-hour horizon is a practical and data-supported choice, as it matches the general driving patterns and charging intervals observed in most electric vehicles.

5.1.3 *Data Imputation*

The objective of this test was to evaluate the performance of the imputation methods discussed in Section 3.1 in terms of how well they generate a continuous dataset suitable for reliable time series forecasting. The imputation methods are also compared against using the raw intermittent data, as processed by the intermittent Transformer model discussed in Section 4.3.3, which can directly handle the irregular intervals without imputation. However, since the intermittent Transformer cannot be applied to multiple stations simultaneously, the IONITY charging station was used as a reference point for this comparison, with the standard Transformer model applied to the datasets using imputation methods.

Ablation study - Data Cleaning	
Data	$\overline{\text{MSE}}$
Fully Cleaned Data	82.42 ± 21.76
- Missing `donken` Data Fix	83.39 ± 21,92
- Faulty `IONITY` Data Fix	84,57 ± 20,95

Table 5: Results of the ablation study comparing the model's performance with different levels of data cleaning on the charging stations in Värnamo. The table shows the MSE values for fully cleaned data, data with missing entries not fixed, and data with faulty entries not corrected. All values are 10^4.

Table 4 presents the MSE for each imputation method across multiple runs. All the imputation methods outperformed the raw data processed by the intermittent Transformer, thereby highlighting the importance of Imputating the dataset for reliable forecasting. The k-NN method showed the lowest average MSE, indicating its effectiveness in creating a smoother and more predictable dataset. This is due to its ability to capture patterns by considering multiple neighbouring points, which reduces data variability and results in smoother time series compared to the LOCF or the NOCB methods. Instead, LOCF and NOCB produce jagged time series with abrupt changes that are harder to predict. In contrast, the k-NN method generates a time series that is much smoother and sufficiently close to the original data, resulting in more accurate occupancy forecasting. Therefore, k-NN will be used for all subsequent tests.

5.1.4 *Data Cleaning*

Table 5 presents the results of an ablation study comparing the model's performance with and without data cleaning. The results indicate that data cleaning provided a slight improvement in the forecasting accuracy, as evidenced by the marginally lower MSE values. While the impact of data cleaning is not substantial, it suggests that removing inconsistencies from the data contributes to more reliable predictions.

5.2 FORECASTING

This section presents the results of forecasting tests conducted to predict EV charging station occupancy using various models. Both traditional methods, such as MA, AR, and ARIMA, alongside advanced DL models such as LSTM, Transformer, and ST-GCN, were evaluated. The experiments aimed to identify the most accurate model for predicting occupancy, with performance assessed based on MSE and the consistency of predictions.

5.2.1 *Experiments*

To compare single-station forecasting methods with multi-station approaches, the IONITY charging station in Värnamo was selected as a reference point. For the multi-station forecasting methods, predictions were generated for the entire Värnamo charging network, and the forecast for the IONITY station was then extracted for comparison with the single-station methods. This location was selected because it is the most used charging station available within the ChargeFinder dataset, resulting in a large range of different sequences and comprehensive data coverage. Its extensive variation in usage patterns allows for a reliable analysis and comparison of forecasting models. Furthermore, the IONITY station has consistently posed the greatest challenge for occupancy prediction during the experiments. This makes it the ideal candidate for evaluating the models in this research project, making it an excellent choice for a comprehensive experiment, as it covers a wide range of sequences and effectively tests the models' performance.

This research project used Optuna, a hyperparameter optimization framework, to determine the best hyperparameters for each model. Optuna is applied to all DL models in this thesis. To ensure the reliability and reproducibility of the results, each model is trained, validated and tested on three different seeds. This approach helps in selecting an optimal set of parameters while mitigating the impact of random variations in the training process, leading to more reliable and consistent results across the models. To provide a more accurate assessment of each model's performance, the evaluation metrics used for comparison are discussed in 3.5.1. The test results are presented in Table 6 and an example of forecasting predictions is shown in 19. It is important to note that all MSE and standard deviation values shown in this section are multiplied by 10^4 for simplicity and ease of comparison in the visualizations.

5.2.2 *MA*

For the MA model, the parameter q was set to 36 to align with the requirement of predicting 36 timesteps into the future. This choice helps prevent the model from producing a constant forecast for the timesteps following $t+q$, and up to $t+L$. To clarify, if a model were set with a q value of 4 and tasked with predicting 10 data points into the future, it would compute the average for the first four data points and then maintain a constant prediction from $t+4$ to $t+10$. Additionally, setting q to 36 was informed by the significant corre-

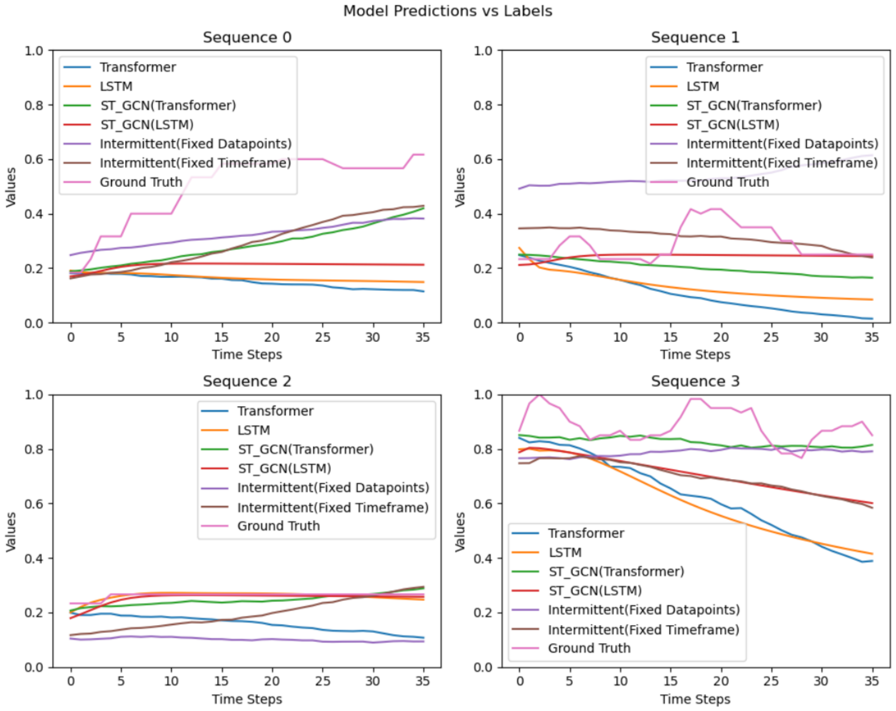


Figure 19: Forecasting predictions generated by various models across four different sequences from the IONITY dataset.

IONITY Data			
Model	**$\overline{\text{MSE}}$**	**Model Type**	**Graph?**
LVCF	252.72 ± 0	Constant	No
MA	392.63 ± 0	Statistical	No
AR	323.18 ± 0	Statistical	No
ARIMA	261.79 ± 0	Statistical	No
LSTM	208.79 ± 6.29	RNN	No
Transformer	168.61 ± 6.15	Transformer	No
Intermittent Transformer (Fixed Datapoints)	304.44 ± 13.02	Transformer	No
Intermittent Transformer (Fixed Timeframe)	240.08 ± 3.29	Transformer	No
ST-GCN (LSTM)	184.38 ± 13.56	RNN	Yes
ST-GCN (Transformer)	142.83 ± 7.04	Transformer	Yes

Table 6: Comparison of the average MSE values and standard deviation on the IONITY dataset for all models employed in this project. All values are multiplied by 10^4.

All Data			
Model	**$\overline{\text{MSE}}$**	**Model Type**	**Graph?**
ST-GCN (LSTM)	97.98 ± 25.79	RNN	Yes
ST-GCN (Transformer)	82.42 ± 21.76	Transformer	Yes

Table 7: Comparison of the average MSE values and standard deviations across all data from all stations and cities for the two ST-GCN models used in this project. All values are multiplied by 10^4.

lations observed between the current time t and the previous $t - 36$ timesteps, as shown in the correlation plot in Figure 9 on page 32.

Overall, the MA model yielded an average MSE value of 392.63 across the dataset, as shown in Table 6. This result is the highest among all the models, indicating the poorest performance in terms of MSE. A likely reason for this outcome is the simplicity of the MA model, which relies solely on past averages and may struggle to capture the underlying temporal patterns and complexities in the data, particularly when the data exhibits non-linear trends or strong seasonality. This limitation likely contributes to the higher errors observed, especially when compared to more sophisticated models capable of better adapting to the non-linearities in the data.

5.2.3 *AR*

Regarding the AR model, the p parameter was set to 36, similar to the q parameter in the MA model. The AR model achieved an average MSE value of 323.18 across the dataset, as shown in Table 6. This performance is an improvement over the MA model, as the AR model can capture more complex relationships within the data. Unlike the MA model, which only considers past averages, the AR model directly uses past observations to inform its predictions. By incorporating these previous values, the AR model is better equipped to detect and model temporal dependencies and patterns, resulting in a lower MSE compared to the MA model.

5.2.4 *ARIMA*

The ARIMA model, which integrates AR, differencing, and MA components, has the parameters p and q for the AR and MA components both set to 36. The differencing parameter d is set to 0, meaning the ARIMA model is applied to the raw, undifferenced dataset, allowing it to capture inherent trends and seasonality. With these parameters, the model achieves an average MSE value of 261.79 on the IONITY dataset, as shown in Table 6. This improvement over both the MA and AR models is expected, as ARIMA combines the strengths of both approaches using past values (AR) and past errors (MA). However, despite these advantages, the ARIMA model still falls short compared to the results achieved by the DL models, which are generally more capable of capturing complex, nonlinear relationships in the data.

As anticipated, traditional ML models demonstrated limited applicability to large time-series datasets. Their primary drawbacks include the inability to capture non-linear relationships, simplistic data modelling, and lengthy runtimes, further compounded by their lack of

GPU utilization. Due to these limitations, DL models such as Transformers, LSTMs, and GCNs have been further explored and optimized in this thesis. These advanced models are better suited to capture complex patterns, trends, and seasonality in the data compared to traditional models like AR, ARIMA, and MA.

5.2.5 *LSTM*

A simple LSTM model, as described in Section 4.3.1, significantly outperformed all the ML models and the horizontal line benchmark, achieving an average MSE score of 208.79, as shown in Table 6. This performance can be attributed to the LSTM's ability to effectively capture long-term dependencies and complex temporal patterns within the data. Unlike traditional ML models, which often struggle with non-linear relationships and time-series data, the LSTM's recurrent architecture allows it to maintain and utilize information across extended time steps. This capability is crucial in accurately predicting EV charging station occupancy, where historical usage patterns and trends play a vital role in forecasting future demand.

When forecasting occupancy, LSTMs tend to produce smooth occupancy curves, as illustrated in Figure 19. This smoothness likely stems from the LSTM's sequential processing, where each prediction is influenced by the preceding ones, resulting in gradual changes in the output. While this can lead to smoother forecasts, the overall trend of the LSTM predictions generally aligns well with actual occupancy, contributing to its better performance compared to the horizontal line and traditional ML methods. However, despite outperforming the ML models, the LSTM was still the worst-performing model among the DL methods.

5.2.6 *Transformer*

The Transformer model achieved the lowest MSE score among all the models designed for single-station prediction, with an MSE of 168.61. This superior performance is likely due to the Transformer's attention mechanism, which enables it to effectively capture both short-term and long-term dependencies across the entire data sequence. Unlike the LSTM model, which processes data sequentially and may struggle with capturing global context, the Transformer processes all time steps in parallel, allowing it to dynamically focus on the most relevant parts of the input sequence. This not only enhances accuracy but also significantly reduces training times. While the LSTM required approximately six minutes per epoch, the Transformer model completed each epoch in just over two minutes—a 67% reduction—making it more efficient for large-scale time-series forecasting tasks.

This efficiency is also evident in the model's occupancy forecast. Because the Transformer can independently attend to specific parts of the sequence, it produces predictions that more accurately reflect abrupt changes or variations in the data, resulting in more detailed and jagged lines in the plots. This capability allows the model to capture short-term fluctuations more effectively than LSTMs, as best demonstrated in sequence 3 of Figure 19.

5.2.7 *Intermittent Transformer*

The Intermittent Transformers are shown in Table 6 and its forecast results in Figure 19. Both the *"fixed amount of data points"* and *"Fixed Length Window"* variants detailed in Section 4.3.3 are shown as *Intermittent(Fixed Datapoints)* and *Intermittent(Fixed Timeframe)*. However, both variants of the Intermittent Transformer performed similarly during testing, with the *Intermittent(Fixed Datapoints)* variant achieving a Mean Squared Error (MSE) of 304.44 and the *Intermittent(Fixed Timeframe)* variant achieving an MSE of 240.08. Despite these results, neither variant achieved predictive performance comparable to the other DL methods. The reduced effectiveness of the Intermittent Transformer could be attributed to its reliance on sparse and irregularly sampled data points, which limits its ability to fully leverage the temporal context. Additionally, the model may have been hindered by a persistent class imbalance, resulting from significant variations in either the input timeframe or the number of data points in the time series.

Unlike the standard Transformer, which excels with continuous data and can dynamically focus on relevant information across the entire sequence, the Intermittent Transformer struggled to analyze patterns and produce accurate results. Although the Intermittent Transformer employs the novel positional encoding technique described in [78] to handle irregular data, it was unable to match the predictive accuracy of more traditional DL models.

5.2.8 *ST-GCN (LSTM)*

The ST-GCN model, which used an LSTM as its temporal component, ranked as the third-best model overall, achieving an MSE of 184.38—an 11.69% improvement over the standalone LSTM. The enhanced performance can be attributed to the integration of multi-station forecasting, which allowed the model to capture not only temporal dynamics within a single station but also spatial dependencies across multiple stations. By leveraging the GCN to model the relationships between different charging stations, the ST-GCN effectively enriched the contextual information available to the LSTM, leading to more informed and accurate predictions. The ability to account for both spatial and temporal factors gave the LSTM-based ST-GCN a distinct advantage over models that considered only one of these dimensions, making it the second-best performer in this comparative analysis.

The average MSE across all charging stations and cities is 97.98 as shown in Table 7. However, as seen in 19, the occupancy predictions produced by the LSTM-based ST-GCN still display the characteristic smoothness typical of regular LSTM models.

5.2.9 *ST-GCN (TRANSFORMER)*

The ST-GCN model that incorporated a Transformer as the temporal element was the best-performing model overall, achieving an MSE of 142.83, as shown in Table 6. The addition of the GCN to the standard Transformer enhanced its performance by 15.29%, mirroring the improvement seen when adding a GCN to the LSTM. This improvement can be attributed to the Transformer's ability to capture long-range dependencies and global context within the data, which, when combined with the GCN's spatial awareness, allowed the model to effectively understand and predict occupancy patterns across multiple stations.

Moreover, as detailed in Table 7, which represents the average MSE across all chargers in the Värnamo network, the Transformer-based ST-GCN outperformed the LSTM-based version in multi-station prediction scenarios with an MSE of 82.42. This superior performance is likely due to the Transformer's parallel processing capabilities, which not only increase computational efficiency but also enhance the model's ability to integrate information from various stations simultaneously, leading to more accurate and robust predictions across the entire network. The Transformer's attention mechanism also allows it to capture localized patterns and variations in the data, resulting in jagged occupancy predictions similar to those produced by the Transformer-only model, as shown in Figure 19.

5.3 TRANSFER LEARNING

This section presents the outcomes of experiments involving the BAM adapter, including comparisons against baseline models using 1%, 5%, 20%, and 50% of the target data. The analysis begins with an overview of the primary experimental results, where the MSE was averaged, and the standard deviation was calculated across three pre-trained seeds, each fine-tuned with three different seeds to ensure reliability. The experiments were conducted in two distinct scenarios: Värnamo to Varberg and Varberg to Malmö, offering an evaluation across different settings by pre-training on one location and fine-tuning on another. This approach provides a thorough assessment of the BAM adapter's performance and adaptability across varying conditions. Moving forward, Värnamo to Varberg will be referred to as Scenario 1, and Varberg to Malmö as Scenario 2 for simplicity.

These results are summarized in Tables 8 and 9, and are visualized through two key plots: MSE vs. the number of trainable parameters, shown in Figures 20 and 21, and MSE histograms for different data percentages, presented in Figures 22 and 23. The results, discussed in detail below, clearly demonstrate that the BAM adapter outperforms the baseline models overall. Despite using significantly fewer parameters, BAM achieves competitive MSE results and even surpasses the baselines in some cases. This highlights the efficiency of the BAM adapter in delivering high performance while reducing computational complexity.

In addition to regular comparisons, an ablation study is subsequently performed to investigate the contribution of the BAM components within the model. The results of this study are presented in both tabular form (Tables 10 and 11) and visualized in several plots: Figures 24, 25, 26, and 27. These experiments provide a comprehensive exploration of the contributions of the different adapter components.

5.3.1 *Comparative Study*

Firstly, by solely investigating Tables 8 and 9, it is easily seen that in Scenario 1, BAM produces good results, beating the other models when utilizing 5% and 50% of the data, and achieving the second highest score with 20% of the data, while delivering a competitive performance with 1%, considering the low number of trainable parameters. For Scenario 2, the BAM adapter produces results that take the third place out of the 8 models tested, across all percentages. At first glance, these results may appear acceptable when considering the numbers alone. In Table 8, a notable observation is the exceptionally high standard deviation for certain models, especially when using only 1% of the data. This is particularly evident in models with no fine-tuning, training from scratch, and fine-tuning only the last GCN layer. One possible explanation for this high variability is that the values are scaled by 10^{-4}, which makes the numbers very small and, in turn, amplifies the relative fluctuations in the results.

Regarding the model without any fine-tuning, the high standard deviation can be attributed to its complete reliance on the initial seed. Since no fine-tuning occurs, the model's performance is heavily influenced by whether the seed is well-aligned with the pre-trained city's data. If the seed happens to closely match the target data, the performance may be decent; if not, the performance can suffer significantly.

The model trained from scratch also exhibits high standard deviation at both 1% and 5% data usage. This is understandable, as with

Värnamo to Varberg					
Model	1%	5%	20%	50%	Parameters
$\mathcal{M}_A \to \mathcal{M}_B$	126.4 ± 99.08	126.4 ± 99.08	126.4 ± 99.08	126.4 ± 99.08	0%
$\mathcal{M}_B \to \mathcal{M}_B$	122.49 ± 28.16	82.59 ± 14.1	55.75 ± 1.73	52.55 ± 2.68	100%
$\mathcal{M}_A \xrightarrow{\text{fine-tune all}} \mathcal{M}_B$	58.37 ± 5.27	54.45 ± 2.21	50.65 ± 1.26	50.2 ± 1.21	100%
$\mathcal{M}_A \xrightarrow{\text{Last GCN}} \mathcal{M}_B$	95.89 ± 49.74	68.25 ± 16.17	57.56 ± 5.50	56.94 ± 6.16	0.22%
$\mathcal{M}_A \xrightarrow{\text{Last Transformer}} \mathcal{M}_B$	62.26 ± 1.05	57.26 ± 4.53	53.17 ± 2.80	53.49 ± 2.74	20.99%
MLP Adapter	71.53 ± 10.70	58.03 ± 3.88	52.44 ± 2.01	51.65 ± 1.53	2.69%
Tiny-Attention Adapter	63.07 ± 5.39	58.24 ± 3.22	52.49 ± 1.59	51.43 ± 2.78	57.01%
BAM	63.14 ± 7.63	53.7 ± 1.81	51.22 ± 1.39	50.09 ± 1.31	3.02%

Table 8: Performance of various models at different data usage percentages (1%, 5%, 20%, and 50%) for the Värnamo to Varberg scenario (i.e., Scenario 1). The "Parameters" column displays the percentage of trainable parameters within each model. All values are multiplied by 10^4.

Varberg to Malmö					
Model	1%	5%	20%	50%	Parameters
$\mathcal{M}_A \to \mathcal{M}_B$	136, 16 ± 10, 63	136, 16 ± 10, 63	136, 16 ± 10, 63	136, 16 ± 10, 63	0%
$\mathcal{M}_B \to \mathcal{M}_B$	138.51 ± 14.26	133.29 ± 16.92	106.28 ± 11.67	99.87 ± 7.57	100%
$\mathcal{M}_A \xrightarrow{\text{fine-tune all}} \mathcal{M}_B$	112, 62 ± 12, 21	107, 04 ± 14, 44	98, 09 ± 4, 95	95, 48 ± 6, 08	100%
$\mathcal{M}_A \xrightarrow{\text{Last GCN}} \mathcal{M}_B$	133, 21 ± 12, 53	125, 27 ± 13, 35	121, 16 ± 13, 58	119, 47 ± 14, 51	0.22%
$\mathcal{M}_A \xrightarrow{\text{Last Transformer}} \mathcal{M}_B$	113, 71 ± 14, 19	109, 47 ± 15, 06	107, 01 ± 13, 61	106, 24 ± 13, 64	20.99%
MLP Adapter	123, 19 ± 14.97	114, 04 ± 16.39	111, 32 ± 17.48	108, 58 ± 14.32	2.69%
Tiny-Attention Adapter	112, 62 ± 13.40	100, 92 ± 4.89	97, 34 ± 4.29	95, 14 ± 4.60	57.01%
BAM	114, 11 ± 12.84	108, 71 ± 15.17	102, 25 ± 9.89	99, 19 ± 7.34	3.02%

Table 9: Performance of various models at different data usage percentages (1%, 5%, 20%, and 50%) for the Varberg to Malmö scenario (i.e., Scenario 2). The "Parameters" column displays the percentage of trainable parameters within each model. All values are multiplied by 10^4.

such limited data, the model's performance is highly dependent on the specific subset of data it encounters during training. With so little data, the model may overfit to specific sequences, leading to poor generalization and, consequently, high variability in performance.

Finally, the "Fine-tune last GCN layer" model shows a consistently high standard deviation across all data percentages. This could be due to the fact that fine-tuning only the last GCN layer may not be sufficient for the model to adapt effectively to the new data. Additionally, with low data percentages, the model may struggle to generalize, resulting in particularly high variability at 1%, and still considerable variation even at 50%. This indicates that fine-tuning just the last layer may not provide enough flexibility for the model to achieve stable performance across different scenarios.

However, upon further investigation, as seen in Figures 20 and 21, the performance of the BAM adapter becomes even more impressive when considering its efficiency in using fewer trainable parameters compared to other models, with the exception of the MLP adapter, which is in a similar range, and the fine-tuning of the last GCN layer, which uses even fewer parameters. Ideally, a model should be as close to the top left corner of the plot as possible, indicating high performance with minimal parameters. With this in mind, it is noteworthy that in scenario 1, the BAM adapter outperforms all other models when using 5% and 50% of the data, secures second place with 20% of the data (just behind Fine-tune All), and ranks fourth on 1% of the data, behind fine-tune All, fine-tune Last Transformer, and Tiny-Attention, which fine-tune 100%, 20.99%, and 57.01% of the model's parameters, respectively. In contrast, BAM achieves these results using only 3.02% of the parameters as trainable.

Nevertheless, when analyzing Figures 20 and 21, the ideal model performance is indicated by positioning in the top left corner, where higher accuracy is achieved with fewer trainable parameters. With this in mind, the BAM adapter's results become more impressive upon closer examination. Despite using significantly fewer trainable parameters compared to other models—except for the MLP adapter, which operates in a similar range, and fine-tuning the last GCN layer, which uses even fewer parameters—the BAM adapter stands out. In scenario 1, it outperforms all other models when using 5% and 50% of the data, secures second place with 20% data (just behind Fine-tune All), and ranks fourth with 1% data. Notably, it achieves these results while only using 3.02% of the parameters, compared to the 100%, 20.99%, and 57.01% used by Fine-tune All, Fine-tune Last Transformer, and Tiny-Attention, respectively.

In addition to Scenario 1, the BAM adapter consistently ranks third across all data percentages in Scenario 2, falling behind Fine-tune All and Tiny-Attention, which utilize significantly more parameters, as mentioned previously. This demonstrates strong performance from BAM, as it achieves a favourable balance between the number of trainable parameters and overall effectiveness. In Scenario 2, it is noteworthy that the two models outperforming BAM are the only ones that fully fine-tune the entire transformer decoder. In contrast, BAM, which ranks third, is implemented as a bridge between the encoder and decoder layers, thereby influencing the decoders. Following BAM, in fourth place, is the model that fine-tunes only the last layer of the transformer, specifically within the decoder. The remaining three models—MLP, fine-tuning only the last GCN layer, and no fine-tuning at all—do not involve fine-tuning the decoder. This highlights that the top four fine-tuning models all involve some degree of fine-tuning the decoder, whereas the bottom three models do not. This suggests that in scenario 2, the decoder has a significant impact on performance. Moreover, this also indicates that the BAM adapter, despite using significantly fewer parameters than the models that placed first, second, and fourth, still exerts enough influence on the decoder to achieve strong performance. This highlights the ef-

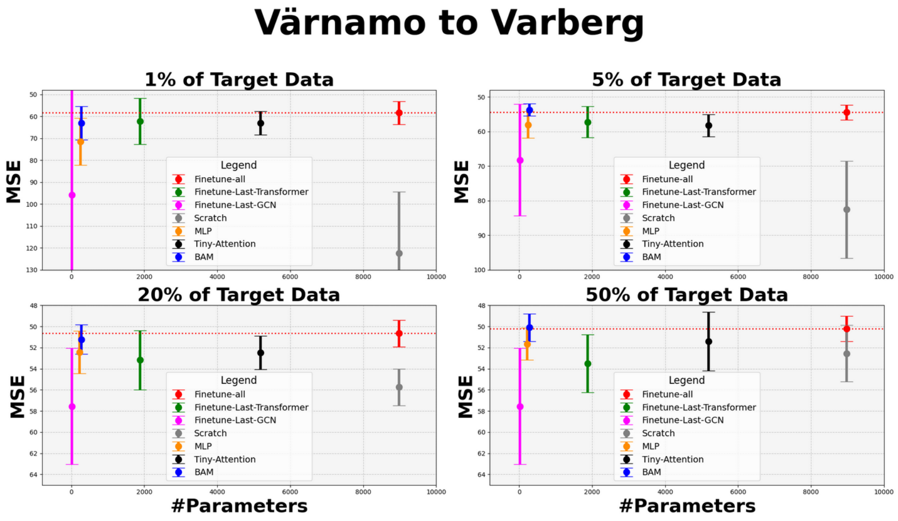


Figure 20: MSE scores for different models at 1%, 5%, 20%, and 50% data usage, displayed across four subplots as a Pareto plot. Each subplot plots the number of trainable parameters on the X-axis and the MSE values on the Y-axis, with error bars representing the standard deviations. Results are shown for the Värnamo to Varberg scenario (i.e., Scenario 1). All MSE values are multiplied by 10^4.

ficiency and effectiveness of the BAM adapter in leveraging limited resources to deliver competitive results.

In addition to these findings, Figures 22 and 23 offer a detailed comparison of the MSE and standard deviation across different models and data utilization percentages. A consistent pattern emerges, showing that the model which fine-tunes only the last GCN layer performs the poorest, with the highest MSE across all data percentages in both scenarios and the highest standard deviation in Scenario 1, as well as one of the worst in Scenario 2.

This poor performance could be due to the fact that fine-tuning only the last GCN layer does not sufficiently capture the complex relationships within the data, as it limits the model's ability to adjust the deeper layers of the network, where more abstract and critical features are learned. Furthermore, the similarity between the pre-trained network and the target network in both scenarios reduces the effectiveness of fine-tuning just the last layer. Additionally, the last GCN layer accounts for only 0.22% of the model's total parameters, which is likely too few to meaningfully impact the model's overall performance and ability to generalize, leading to higher errors and greater variability in its predictions.

Further analysis reveals a consistent decrease in MSE across all models in both scenarios as more data is utilized. In terms of standard deviation, Scenario 1 shows a clear reduction as data usage increases. However, in Scenario 2, this decreasing trend in standard deviation is observed only in the "Fine-tune All," "Tiny-Attention Adapter," and "BAM" models. This pattern supports the assumption that models

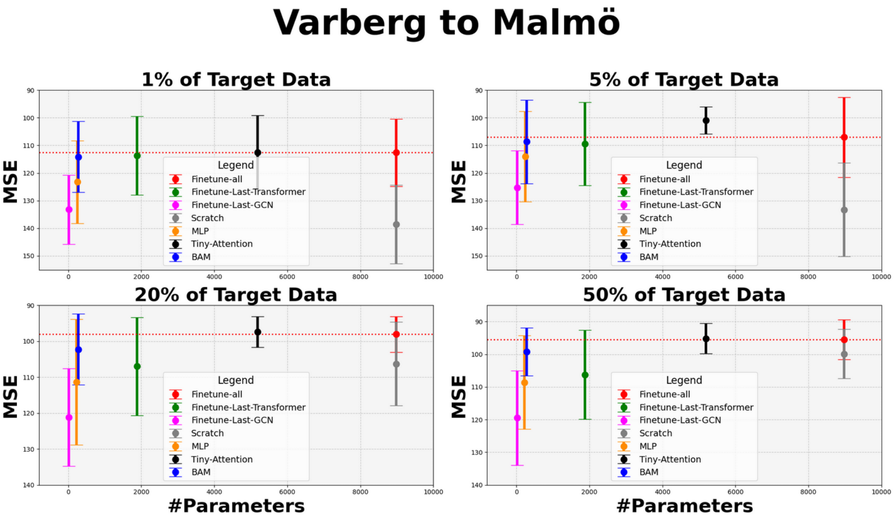


Figure 21: MSE scores for different models at 1%, 5%, 20%, and 50% data usage, displayed across four subplots as a Pareto plot. Each subplot plots the number of trainable parameters on the X-axis and the MSE values on the Y-axis, with error bars representing the standard deviations. Results are shown for the Varberg to Malmö scenario (i.e., Scenario 2). All values are multiplied by 10^4.

with the capability to fine-tune or modify the Transformer's decoders tend to achieve better performance.

Moreover, in Scenario 2 and across most data percentages in Scenario 1, it is evident that the original MLP adapter underperforms compared to the BAM and Tiny-Attention adapters. This strongly suggests that the absence of attention mechanisms, particularly the lack of global context in the sequence, negatively impacts the effectiveness of fine-tuning on these models and this dataset.

Finally, it is evident that the model trained from scratch, utilizing 100% of the parameters, only becomes competitive at the 20% and 50% data usage levels. This provides strong evidence that fine-tuning has a significant positive impact, as even with as little as 1% of fine-tuning data, a model can outperform one trained entirely from scratch. However, as the percentage of fine-tuning data increases, the difference in performance between fine-tuning and training from scratch diminishes, as illustrated in Figures 20 and 21.

5.3.2 *Top-down Ablation Study*

The Top-down ablation study was conducted to analyze the impact of each individual component of the BAM adapter proposed in this thesis. By systematically removing or altering key elements, it becomes possible to identify which components are essential for achieving optimal results. This analysis not only highlights the architectural con-

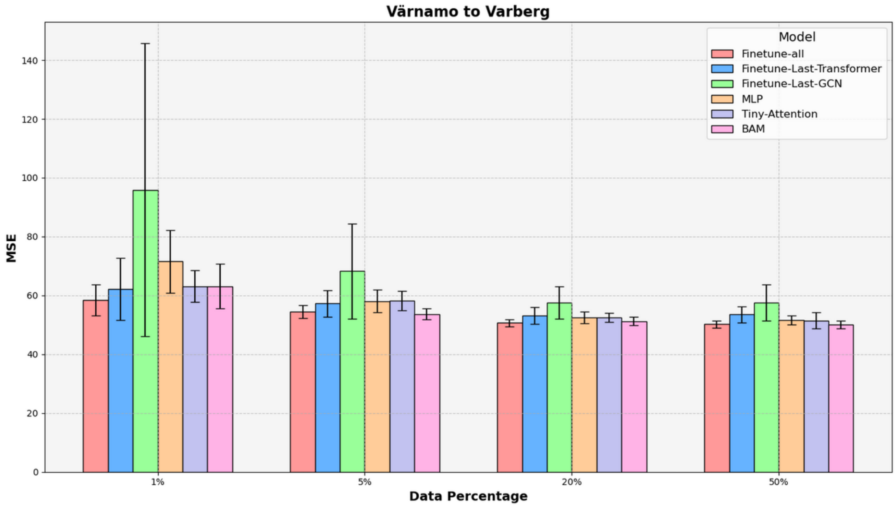


Figure 22: Overview of the diverse models' MSE performance and standard deviation across 1%, 5%, 20%, and 50% data usage in the Värnamo to Varberg scenario (i.e., Scenario 1). All values are multiplied by 10^4.

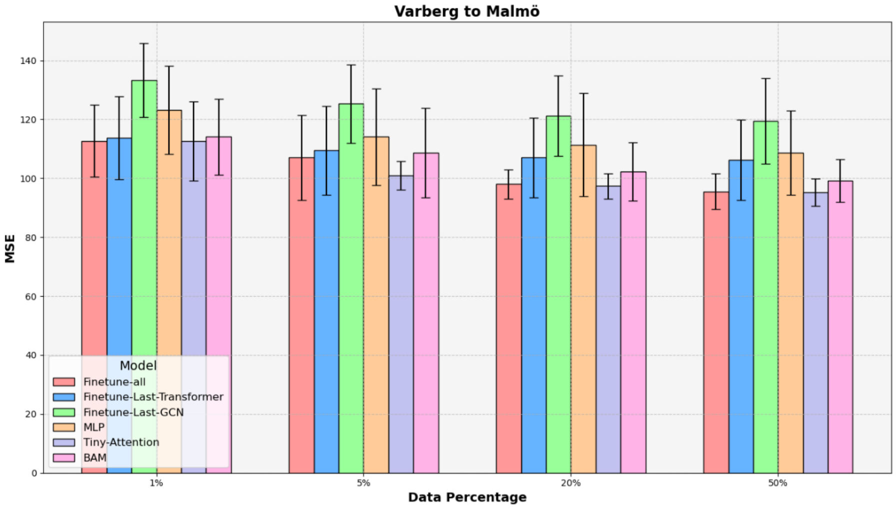


Figure 23: Overview of the diverse models' MSE performance and standard deviation across 1%, 5%, 20%, and 50% data usage in the Varberg to Malmö scenario (i.e., Scenario 2). All values are multiplied by 10^4.

Top-down ablation study (Värnamo to Varberg)				
Model	1%	5%	20%	50%
BAM	63.14 ± 7.63	53.7 ± 1.81	51.22 ± 1.39	50.09 ± 1.31
- GCN Fine-tuning	65, 28 ± 9, 98	55, 46 ± 1, 90	52, 04 ± 1, 43	50, 51 ± 1, 25
- Bridge	55, 39 ± 3, 86	54, 53 ± 1, 91	53, 23 ± 1, 10	51, 77 ± 1, 63
- Transformer Bridge	67, 03± 7, 12	57, 67 ± 2, 84	53, 83 ± 2, 97	52, 41 ± 2, 16
- Attention	61, 55 ± 7, 33	54, 42 ± 2, 01	52, 54 ± 1, 33	51, 18 ± 1, 44
- Dropout	64, 77 ± 7, 54	54, 88 ± 2.05	51, 30 ± 1, 44	50, 53 ± 1, 39
- Layer normalization	61, 47 ± 9, 36	54, 80 ± 2, 17	52, 79 ± 1, 36	51, 46 ± 1, 62
- Scalar	65, 82± 9, 37	55, 45 ± 1, 93	52, 35 ± 1, 22	50, 21 ± 2, 34

Table 10: Top-down ablation study on this study's proposed BAM for the Värnamo to Varberg scenario (i.e., Scenario 1). All values are multiplied by 10^4.

tributions of the model but also provides insights into potential areas for further improvement.

The results, presented in Tables 10 and 11, suggest that in Scenario 2, removing any component results in an increase in MSE, indicating that all components are necessary to achieve the optimal performance with the BAM adapter. In Scenario 1, while most components generally contribute positively, there is an exception at the 1% data usage level. Specifically, at 1% in Scenario 1, the BAM adapter performs better without the bridge between the GCN and Transformers, as well as without Attention Layer normalization. As previously explained, attention mechanisms are known to be data-hungry, which may explain why performance improves without attention in Scenario 1 when only 1% of the data is used. As data usage increases to 5% and beyond, attention begins to contribute to better performance.

The performance differences between the scenarios suggest several possible explanations. In Scenario 1, the patterns and features learned from Värnamo may not align closely with those in Varberg, causing the attention component to overfit by amplifying irrelevant features. Another possibility is that the 1% of training data in Scenario 1 is not highly generalizable; it may disproportionately represent a specific trend in the dataset, leading the model to overfit and perform worse on validation and test data.

A similar reasoning applies to the bridging between the GCN and the Transformer. In Scenario 1, the model performs better without the bridge, while in Scenario 2, it performs better with it. This could be due to the differences in data trends, where the attention mechanism within the bridge might highlight irrelevant time steps, as learned in Värnamo when applied to Varberg. Additionally, the selected 1% of data may not be sufficient to generalize the transformation of GCN features, limiting their effective preparation for the Transformers.

Turning to the Pareto plots in Figures 24 and 25, one outlier stands out across both scenarios and all data percentages: the BAM model

Top-down ablation study (Varberg to Malmö)				
Model	1%	5%	20%	50%
BAM	114.11 ± 12.84	108.71 ± 15.17	102.25 ± 9.89	99.19 ± 7.34
- GCN Fine-tuning	121.55 ± 11.46	113.98 ± 12.66	107.03 ± 10.21	103.01 ± 8.64
- Bridge	118.35 ± 18.70	113.31 ± 18.02	105.99 ± 11.61	101.96 ± 7.81
- Transformer Bridge	120.38 ± 11.47	114.11 ± 12.48	111.58 ± 11.63	109.03 ± 11.26
- Attention	119.14 ± 13.77	111.62 ± 16.72	104.56 ± 11.99	100.70 ± 8.98
- Dropout	120.34 ± 11.63	110.85 ± 15.80	105.26 ± 11.07	100.03 ± 9.43
- Layer normalization	121.98 ± 12.12	112.54 ± 14.80	105.47 ± 9.82	101.65 ± 7.88
- Scalar	119.78 ± 13.26	111.67 ± 14.89	105.30 ± 10.16	101.50 ± 8.30

Table 11: Top-down ablation study on this study's proposed BAM for the Varberg to Malmö scenario (i.e., Scenario 2). All values are multiplied by 10^4.

without the adapter placed inside the Transformer(Magenta). This adapter, which bridges the encoders and connects the last encoder with the decoders, results in the greatest performance loss when removed. Notably, this component also has the highest number of trainable parameters. Another observation is that dropout has minimal impact on standard deviation but provides a noticeable improvement in MSE. Despite requiring no additional parameters in the adapter, the dropout mechanism makes it a highly parameter-efficient component. In the context of transfer learning, particularly when fine-tuning with limited data (1%, 5%, 20%, and 50%), dropout plays a crucial role by helping the model generalize better to the target domain. It does this by reducing reliance on any specific subset of features, which is particularly beneficial when dealing with smaller amounts of data. This regularization effect prevents overfitting to the source domain features and encourages the model to adapt more effectively to the target data, thereby enhancing the overall performance of BAM across different fine-tuning scenarios. Additionally, the scalar component also demonstrates high parameter efficiency, offering considerable improvement while utilizing very few parameters.

To summarize, an analysis of Figures 24 and 25 reveals that, on average, the loss in performance and reduction in trainable parameters follow a clear linear pattern. This consistency underscores the importance of including all components in the BAM model.

Furthermore, Figures 26 and 27, show a clear overview of the performance of BAM and its ablated compontents in different scenarios. In Scenario 1, the performance improves (lower MSE) as more data is used, with diminishing improvements at higher data percentages, as expected. Notably, the 1% data usage performs better without attention, the bridge, or layer normalization, but these components become beneficial at 5% data usage and above, consistent with previous discussions. Additionally, without the Transformer bridge, it con-

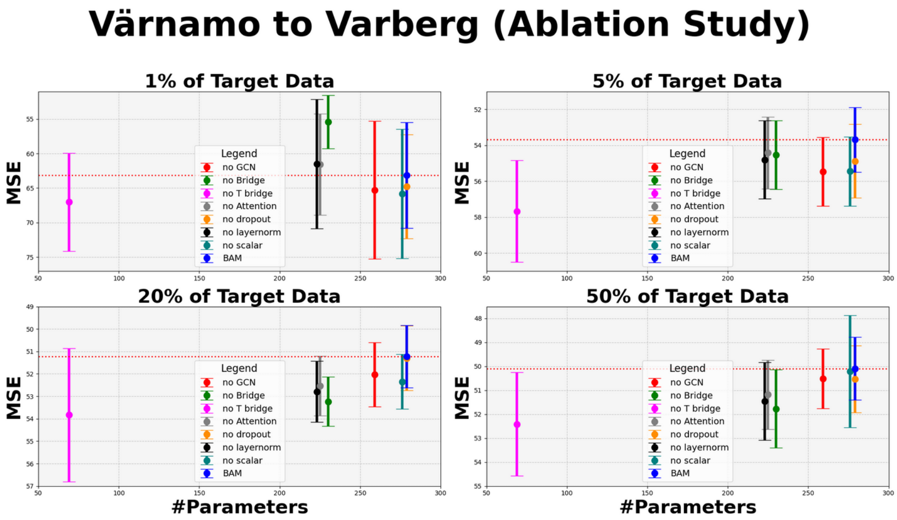


Figure 24: MSE scores for Top-down ablation study at 1%, 5%, 20%, and 50% data usage, displayed across four subplots as a Parito plot. Each subplot plots the number of trainable parameters on the X-axis and the MSE values on the Y-axis, with error bars representing the standard deviations. Results are shown for the Värnamo to Varberg scenario (i.e., Scenario 1). All values are multiplied by 10^4.

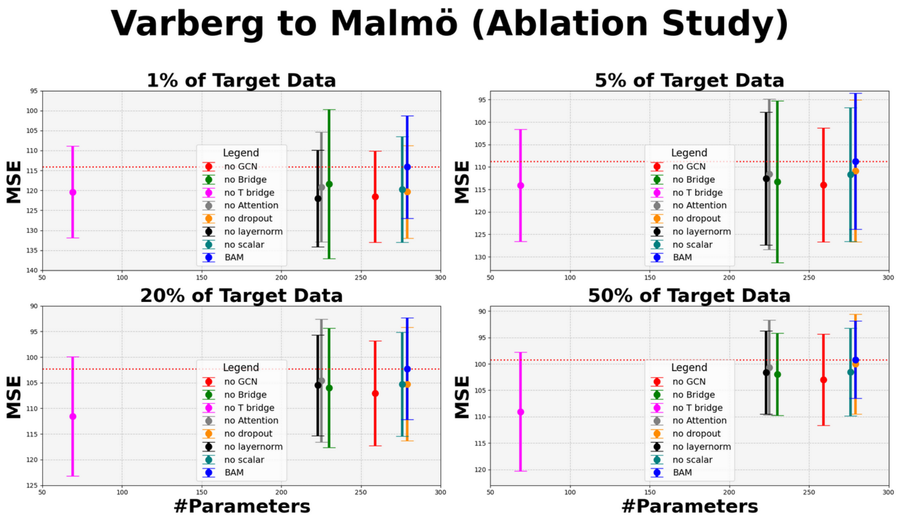


Figure 25: MSE scores for Top-down ablation study at 1%, 5%, 20%, and 50% data usage, displayed across four subplots as a Parito plot. Each subplot plots the number of trainable parameters on the X-axis and the MSE values on the Y-axis, with error bars representing the standard deviations. Results are shown for the Varberg to Malmö scenario (i.e., Scenario 2). All values are multiplied by 10^4.

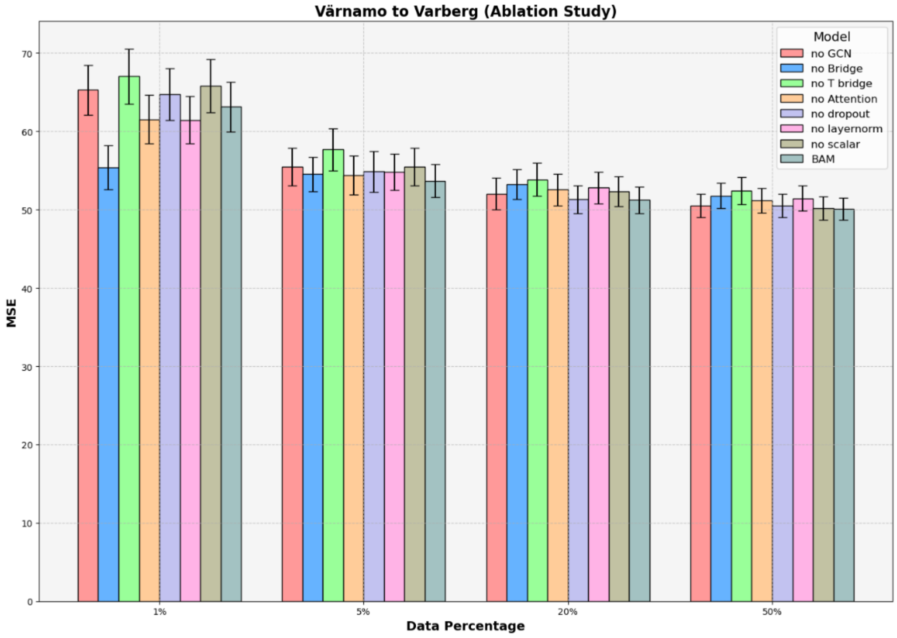


Figure 26: Overview of the MSE performance and standard deviation in Top-down ablation study across 1%, 5%, 20%, and 50% data usage in the Värnamo to Varberg scenario (i.e., Scenario 1). All values are multiplied by 10^4.

sistently shows the poorest performance across all data percentages, reaffirming earlier observations.

Figures 26 and 27 further confirm the findings discussed earlier, particularly those highlighted in the Pareto plots. In Scenario 1, the performance improves (lower MSE) with increased data usage, as expected, although the rate of improvement diminishes at higher data percentages. A notable new finding is that with only 1% of the data, the model performs better without attention, the bridge, or layer normalization. However, these components become increasingly beneficial as the data usage reaches 5% and above, demonstrating their importance in scenarios with more data. Additionally, the model without the Transformer bridge consistently shows the poorest performance across all data percentages, highlighting the critical role this component plays in the overall effectiveness of the BAM adapter. These insights provide a deeper understanding of how different components contribute to model performance under varying data conditions.

In Scenario 2, BAM without the Transformer bridge once again performs the worst, showing minimal improvement in MSE, with standard deviation remaining nearly constant across all data percentages. The bridge exhibits the highest standard deviation at 1% and 5% data usage, but it stabilizes at higher percentages, aligning with the other versions' standard deviations. This suggests that the bridge plays a role in stabilizing performance at lower data usage levels.

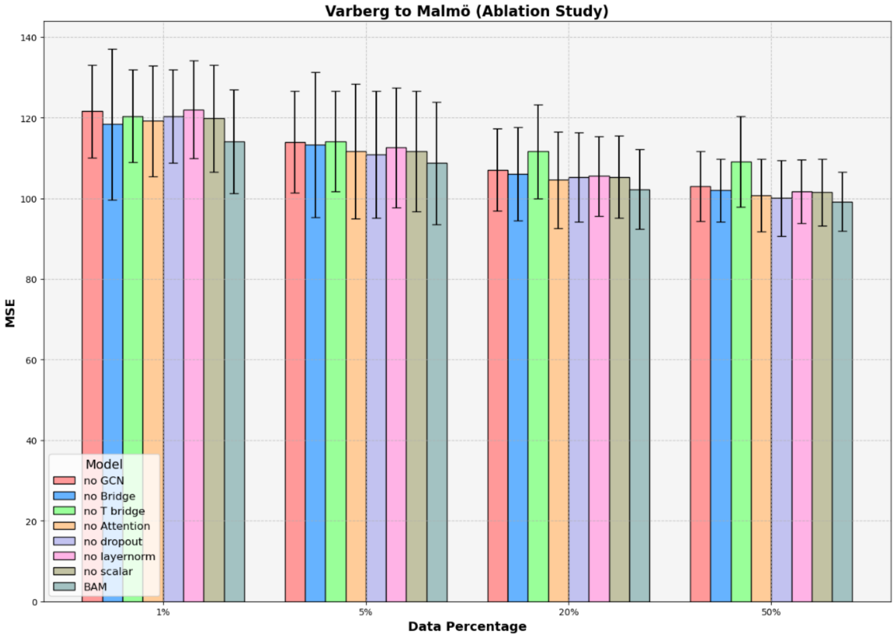


Figure 27: Overview of the MSE performance and standard deviation in Top-down ablation study across 1%, 5%, 20%, and 50% data usage in the Varberg to Malmö scenario (i.e., Scenario 2). All values are multiplied by 10^4.

An interesting observation in this scenario is that the standard deviation increases at 5% data usage for every BAM version compared to 1%. This may be due to the model being exposed to more information at 5%, enabling it to learn more complex patterns. However, the limited amount of data can still lead to overfitting to specific examples or noise, resulting in greater variability in performance and, consequently, a higher standard deviation. As data usage increases to 20% and 50%, the model benefits from a more diverse and representative dataset, which mitigates overfitting and promotes more stable learning. This stability is reflected in the decreasing standard deviation, indicating more consistent performance across different runs.

Bottom-up ablation study (Värnamo to Varberg)				
Model	1%	5%	20%	50%
+ Bottleneck	79,53 ± 13,07	74,54 ± 8,11	67,96 ± 3,45	63,79 ± 2,36
+ 1-head Attention	77,51 ± 12,22	64,39 ± 7,98	57,55 ± 3,21	56,42 ± 2,05
+ Dropout	74,74 ± 11,62	62,96 ± 7,21	56,34 ± 3,04	56,33 ± 2,02
+ Skip Connection	69,23 ± 10,13	58,93 ± 1,93	56,27 ± 1,61	55,34 ± 1,84
+ Layer normalization	68,36 ± 10,03	57,97 ± 1,54	55,51 ± 1,55	54,52 ± 1,59
+ Scalar	66,04 ± 8,48	57,32 ± 2,07	53,89 ± 1,48	52,95 ± 1,30
+ GCN (=BAM)	63.14 ± 7.63	53.7 ± 1.81	51.22 ± 1.39	50.09 ± 1.31

Table 12: Bottom-up ablation study of the various stages of this study's proposed BAM for the Värnamo to Varberg scenario (i.e., Scenario 1), when being built from the bottom up. All values are multiplied by 10^4.

Bottom-up ablation study (Varberg to Malmö)				
Model	1%	5%	20%	50%
+ Bottleneck	141,22 ± 27,53	125,37 ± 21,55	121,78 ± 17,53	118,34 ± 15,72
+ 1-head Attention	124,20 ± 16,84	116,08 ± 14,63	113,66 ± 15,34	112,76 ± 15,04
+ Dropout	122,96 ± 16,67	114,92 ± 14,48	112,52 ± 15,19	111,63 ± 14,89
+ Skip Connection	117,35 ± 14,55	113,74 ± 16,80	107,86 ± 15,75	107,25 ± 16,55
+ Layer normalization	117,22 ± 13,64	113,66 ± 15,97	107,32 ± 11,23	106,71 ± 9,45
+ Scalar	116,91 ± 13,40	113,25 ± 15,58	106,62 ± 11,13	105,19 ± 9,53
+ GCN (=BAM)	114,11 ± 12,84	108,71 ± 15,17	102,25 ± 9,89	99,19 ± 7,34

Table 13: Bottom-up ablation study of the various stages of this study's proposed BAM for the Varberg to Malmö scenario (i.e., Scenario 2), when being built from the bottom up. All values are multiplied by 10^4.

5.3.3 *Bottom-up ablation study*

In this section, a Bottom-up ablation study is presented and discussed. Unlike Top-down ablation study, which analyzed the final architecture of the BAM adapter by systematically removing components to evaluate their impact, Bottom-up ablation study takes a bottom-up approach. The BAM adapter is built from its most basic form, with components added one by one, providing insight into how each piece contributes to the overall performance. This process not only highlights the incremental impact of each part but also illustrates the reasoning behind the model's design. By gradually constructing the model, this study demonstrates how individual components interact and contribute to the final architecture

Beginning with the Bottom-up ablation analysis, Tables 12 and 13 present results across all model configurations, starting with the ***bottleneck layer***, a common component in many adapter architectures due to its ability to compress information and reduce the number of trainable parameters. This initial setup performed well, particularly on smaller datasets, demonstrating its effectiveness in early training. However, as the dataset size increased, the performance stabilized, indicating that while the bottleneck layer could manage simpler patterns, it lacked the capacity to generalize to more complex data. This highlighted the need for additional mechanisms to improve the model's ability to handle larger and more intricate datasets.

To address the limitations of the bottleneck layer, a ***1-head attention*** mechanism was introduced. Attention mechanisms are well known for their ability to capture intricate dependencies, and were expected to improve the model's performance, particularly with larger datasets. A single attention head was chosen specifically to keep the number of trainable parameters as low as possible within the adapter, balancing complexity with efficiency. This addition proved effective, as the 1-head attention consistently outperformed the bottleneck configuration, especially as data volume increased. By focusing on critical temporal and spatial features, the attention mechanism enabled the model to handle more complex patterns that the bottleneck alone could not manage.

Given the challenges posed by small datasets, where overfitting is a common issue, ***dropout*** was introduced to further stabilize training. Dropout randomly disables neurons during training, forcing the model to learn broader, more generalizable features rather than memorizing specific data points. This was particularly effective in the early stages of training, where data scarcity leads to overfitting. The introduction of dropout significantly enhanced the model's generalization ability across all data sizes, with a more pronounced impact with 1% and 5% of data. This outcome confirms the importance of dropout in data-sparse scenarios, where preventing overfitting is crucial.

With deeper networks, challenges such as vanishing gradients and unstable training dynamics become more pronounced. To mitigate these issues, ***skip connections*** were incorporated. These connections allowed gradients to flow more directly through the network, stabilizing the learning process, especially in deeper architectures. Skip connections also preserved important features from the original input to the BAM adapter, which proved crucial when dealing with limited data. With 1% and 5% of data, skip connections played a key role in maintaining critical information, thereby enhancing the model's ability to generalize from limited data.

Next, ***layer normalization*** was introduced to normalize inputs across features, leading to more stable training and faster convergence, helping to smooth out the training process across varying data sizes. Layer normalization was added specifically because, when BAM is placed between the GCN and the Transformer, it bridges components that lack layer normalization. Similarly, when BAM is placed inside the Transformer, bridging each Encoder-layer with the subsequent layer, it is placed outside of the existing layer normalization layers in the Transformer. Therefore, BAM requires its own layer normalization to ensure stable performance. The most notable effects of layer normalization were observed at the 20% and 50% data levels, where it reduced performance variability, as evidenced by the lower standard deviation, and ensured more consistent results. While its impact was less pronounced at the 5% data level, this behavior aligns with earlier findings from the Top-down ablation study, where standard deviations were generally higher at 5% but improved as more data became available.

A ***scalar*** addition was incorporated at the end of the BAM adapter as a final step to adjust the features by appropriately scaling them before outputting. Although its impact was more modest compared to other components, it contributed to improving the model's stability and forecasting. This relatively minor effect was expected, as scalar adjustments involve very few parameters and are intended more as a subtle refinement rather than a significant transformation of the model architecture.

The final addition that compleated the BAM configuration was the addition of ***fine-tuning the last layer of the GCN*** when the adapter is placed between the GCN and Transformer, in order to also take spatial aspect into account and that barely introduced more trainable parameters. The BAM is designed to bridge the GCN and the Transformer within the ST-GCN model by creating a much smoother flow of information (features) between the spatial and temporal components in the ST-GCN. BAM significantly outperformed the other configurations, particularly due to its ability to effectively manage both spatial and temporal information. By fine-tuning the last layer of the GCN, BAM optimized the flow of data across the model. Placing BAM between the GCN and the Transformer, ensured that the sequence-wise attention mechanism could fully prepare the GCN's output for the Transformer's temporal processing, enhancing both performance and generalization across data scenarios. Additionally, BAM was strategically placed on top of the encoder layers, following the Transformers pre-trained layer normalization to ensure that learned transformations were preserved. This design choice, preserved essential information while maintaining the benefits of the BAM adapter.

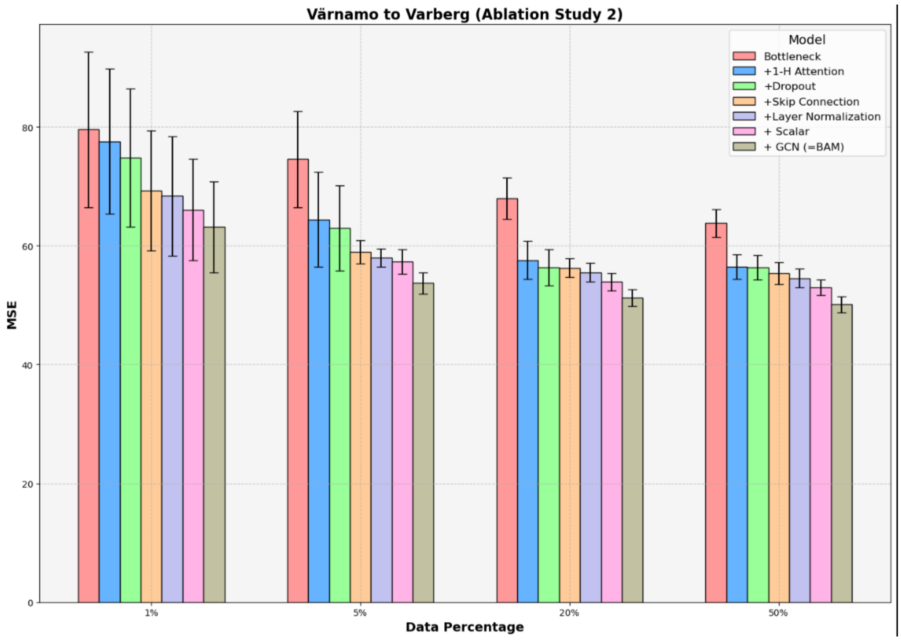


Figure 28: Overview of the MSE performance and standard deviation in Bottom-up ablation study across 1%, 5%, 20%, and 50% data usage in the Värnamo to Varberg scenario (i.e., Scenario 1). All values are multiplied by 10^4.

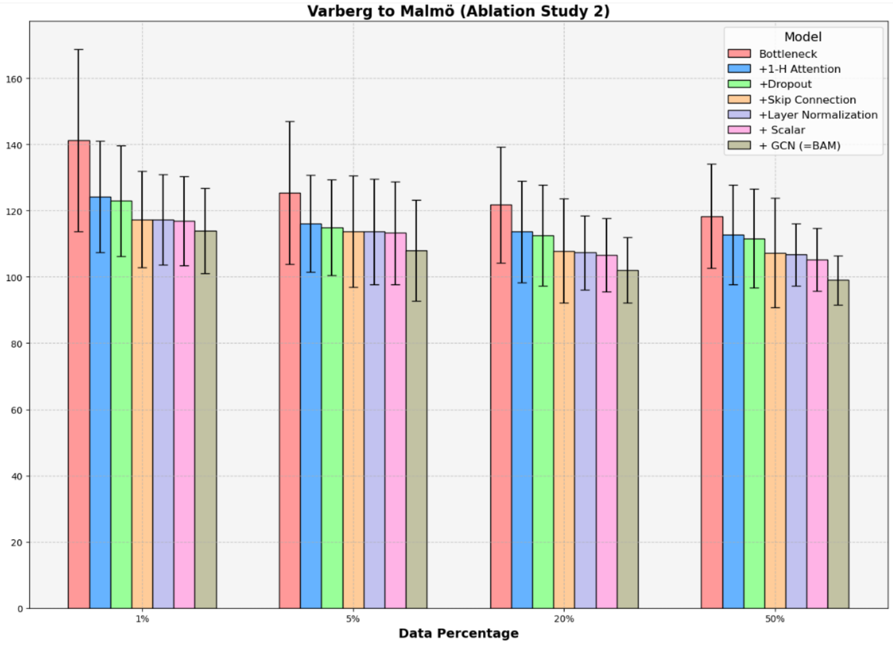


Figure 29: Overview of the MSE performance and standard deviation in Bottom-up ablation study across 1%, 5%, 20%, and 50% data usage in the Varberg to Malmö scenario (i.e., Scenario 2). All values are multiplied by 10^4.

6 THESIS CONTRIBUTIONS

This chapter outlines the key contributions made in this thesis, focusing on the methodologies and techniques developed to enhance the accuracy and efficiency of EV charging station occupancy forecasting. It covers the critical aspects of data pre-processing, model optimization, and transfer learning, as well as the limitations encountered during the research.

6.1 PRE-PROCESSING

The first research question of this study aims to identify the most effective data cleaning, transformation, and imputation techniques for addressing missing values, outliers, and data inconsistencies in the ChargeFinder dataset. Through analysis and experimentation, several key techniques were identified as particularly useful:

- **Normalization of occupancy data**: To account for variations in the number of charging outlets available at different stations over time, the occupancy data was normalized by recalculating the occupancy as the ratio of `occupied_count` to `outlet_count`. This normalization ensured that the occupancy metrics were consistent and comparable across different stations and time periods, improving the reliability of the forecasting model.
- **Data Resampling with Imputation Methods**: The ChargeFinder dataset exhibited intermittent time series data with gaps due to varying querying activity. Initially, an attempt was made to directly use this intermittent data in the model, but it did not yield satisfactory results. Consequently, various imputation methods were tested, with k-NN emerging as the most effective approach for converting the intermittent time series into a continuous format. This method considers the relationships between data points to estimate missing values, thereby preserving the underlying patterns in the data and improving the model's ability to learn from historical trends.
- **Handling outliers and anomalies**: The dataset was carefully examined for outliers and anomalies, particularly in the `outlet_count` and `occupancy_count` features. For example, unexpected fluctuations in the `outlet_count` at certain stations were corrected by capping the values at a reasonable maximum,

  The data was also supplemented with the auxiliary data discussed in Section 4.1.2. Initial tests of the prediction models using this auxiliary data showed poor results. However, this approach was not fully explored due to time and resource constraints. As a result, the potential benefits of integrating these additional features remain unclear, leaving this as an area for future work. Further investigation into the use of these multivariate features could provide valuable insights and potentially improve the accuracy of occupancy predictions.

In summary, the combination of normalization, k-NN imputation and outlier correction enhanced the quality of the ChargeFinder dataset. These data preparation steps were essential in improving the accuracy of the ST-GCN model used for forecasting EV charging station occupancy.

6.2 optimizing forecasting models

To effectively minimize the MSE when predicting EV charging station occupancy over the next three hours using the ChargeFinder dataset, we focused on developing and optimizing an ST-GCN model. This model is specifically designed to capture both spatial and temporal dependencies within the dataset. The ST-GCN model was selected for its ability to process graph-structured data, capturing the intricate relationships between multiple charging stations while simultaneously accounting for temporal dynamics. By integrating graph convolutional layers with temporal layers as described in Section 4.3.5, our model can predict occupancy levels with heightened accuracy. This design leverages the inherent structure of the ChargeFinder dataset, where spatial interdependence between nearby charging stations influences occupancy trends.

To further optimize the model, we tune the models' hyperparameters using Optuna. This allowed us to systematically explore a broad range of hyperparameters, focusing on those that most significantly impacted the model's performance. The results were measured against the primary metric, MSE, and through iterative adjustments, we were able to minimize prediction errors effectively. The final ST-GCN model demonstrated a significant reduction in MSE compared to baseline models, such as traditional ARIMA and LSTM networks, by predicting occupancy with a high degree of accuracy over the crucial three-hour window.

6.3 TRANSFER LEARNING TECHNIQUE

To implement and optimize a transfer learning technique that minimizes the number of trainable parameters while maintaining competitive MSE performance for predicting EV charging station occupancy, the BAM adapter proposed in this study demonstrates its effectiveness as a solution. The BAM adapter achieves a balance between reducing the number of parameters and preserving model performance by carefully combining dimensionality reduction, attention, non-linear transformations, and residual connections.

The results indicate that the BAM adapter performs well across various scenarios and data percentages while using significantly fewer trainable parameters compared to other baseline models. For example, in Scenario 1, the BAM adapter outperformed other models, especially at 5% and 50% data usage, all while using only 3.02% of the model parameters. Even in the more complex Scenario 2, the BAM adapter consistently ranked third, maintaining competitiveness despite using fewer parameters.

Furthermore, two extensive ablation studies were conducted to evaluate the contributions of each component within the BAM adapter architecture. In the Top-down ablation study, each component of the BAM adapter was iteratively removed and then reintroduced, examining its individual impact on performance. The Transformer bridge within the adapter emerged as particularly crucial, significantly stabilizing and improving MSE performance. This result highlights that even partial fine-tuning, when applied effectively, can substantially enhance model accuracy.

In the Bottom-Up ablation study, a bottom-up approach was taken, starting with the most basic components and progressively building toward the final architecture. This method not only demonstrated the effectiveness of each component but also validated the rationale behind their inclusion. The results revealed a clear trend: with each additional component, both MSE and standard deviation consistently improved, confirming the thoughtful design choices behind the architecture.

6.4 LIMITATIONS

This chapter addresses the limitations of the approach, data, and models used in this research. One key limitation lies in the quality and scope of the dataset features. The available features may not fully capture the complexity of the task, potentially limiting the model's performance and generalizability. For instance, the dataset lacked features such as pricing, which could have provided deeper insights into station behaviour and contributed to more stable model outcomes.

Furthermore, the BAM adapter, while effective in the tested scenarios, has specific limitations regarding its application. The model was only validated with the current dataset, raising concerns about its generalizability to other, potentially more, or less, complex scenarios. Additionally, the approach assumes prior knowledge of the number of nodes or stations to be considered, which limits its flexibility. Transfer learning, as applied in this study, is constrained to the same number of stations and up to a maximum of three hours, which may not be applicable in broader contexts.

7 CONCLUSION

7.1 SUMMARY

As EV adoption rises, the need for reliable charging infrastructure grows. A key challenge is accurately forecasting charging station occupancy to optimize usage and reduce waiting time. However, traditional methods often demand extensive data and computational resources, particularly when scaling across multiple stations with varying data availability. Therefore, transfer learning, which adapts models trained on one task to another with minimal data, addresses this challenge, enhancing the scalability and efficiency of occupancy forecasting across diverse charging networks.

A critical first step in this research was ensuring the consistency and reliability of the data used for model training. The ChargeFinder dataset presented challenges such as varying outlet counts over time and intermittent time series data. To address these, the study implemented a normalization strategy, calculating the ratio of occupancy count to outlet count, which proved crucial for maintaining consistency across different time periods. Furthermore, the use of k-NN for data imputation effectively converted intermittent time series data into a continuous format, preserving the underlying patterns. Outliers and anomalies, such as erroneous outlet counts, were also systematically addressed to ensure the reliability of the data, forming a strong foundation.

Building on the pre-processed data, the research focused on developing a reliable forecasting model using a transformer-based ST-GCN. This model was chosen for its ability to capture both spatial and temporal dependencies, enhancing the ability to forecast the occupancy of EV charging stations across multiple locations. Through hyperparameter tuning with the Optuna framework, the ST-GCN model was optimized, leading to a significant reduction in MSE. In addition, the ST-GCN also outperformed traditional baselines like ARIMA and LSTM. The combination of the ST-GCN model and the refined preprocessing techniques, particularly k-NN imputation, resulted in accurate occupancy predictions, making it a valuable tool for managing EV charging infrastructure.

To further improve the model's applicability, particularly in data-sparse scenarios, we introduced the BAM adapter as a transfer learning solution. BAM was designed to allow the ST-GCN model to be fine-tuned efficiently across different city networks with limited additional data. The experiments showed that BAM consistently results in strong performance across various data usage scenarios, often outperforming or closely matching other models in terms of MSE, while using significantly fewer parameters. Remarkably, only 3.02% of the total parameters needed to be trainable, making BAM a highly efficient solution compared to existing ones. The effectiveness of BAM was further validated through two ablation studies, highlighting the rationale behind each component, its contributions, and the steps leading to the final architecture. These studies demonstrated the model's ability to achieve competitive results, even with limited computational resources. This demonstrates BAM's potential as a powerful tool for scaling EV charging station occupancy forecasting across diverse and resource-limited environments.

7.2 FUTURE WORK

- **Enhancing Dataset Features:** The current dataset from ChargeFinder lacks certain features, such as pricing information, which could provide deeper insights into charging station behaviour. Future work could focus on incorporating additional relevant features to better capture the complexity of the task and improve the model's performance and generalizability.
- **Expanding BAM Adapter Validation:** While the BAM adapter demonstrated effectiveness in the scenarios tested, its generalizability to other datasets and more complex scenarios remains unproven. Future work should focus on validating the BAM adapter across a wider range of datasets and contexts, including those with different numbers of nodes or stations, to assess its broader applicability.
- **Increasing Flexibility in Transfer Learning:** The transfer learning approach in this study was limited by the assumption of a fixed number of stations and a maximum forecasting horizon of three hours. Future research could explore methods to increase the flexibility of transfer learning models, making them applicable to a broader range of scenarios, including varying numbers of stations and longer forecasting horizons.
- **Real-Time System Integration:** Future research could focus on integrating the model into real-time systems to assess its performance in dynamic environments. Evaluat-

ing how well the model supports rapid decision-making in such settings would provide valuable insights into its practical applications.

[95] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. URL https://arxiv.org/abs/1902.00751.

[96] Hongyu Zhao, Hao Tan, and Hongyuan Mei. Tiny-attention adapter: Contexts are more important than the number of parameters. *arXiv preprint arXiv:2211.01979*, 2022.

[1] IEA. Electric Vehicles, 2023. URL https://www.iea.org/energy-system/transport/electric-vehicles. Accessed on 2023-11-24.

[2] European Union. The European Green Deal, 11 December 2019. URL https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_en. Accessed on 2023-11-24.

[3] European Union. Zero emission vehicles: first 'Fit for 55' deal will end the sale of new CO2 emitting cars in Europe by 2035, 28 October 2022. URL https://ec.europa.eu/commission/presscorner/detail/en/IP_22_6462. Accessed on 2023-11-24.

[4] Volvo Cars. Volvo cars to be fully electric by 2030. Press release, 2021. URL https://www.media.volvocars.com/global/en-gb/media/pressreleases/277409/volvo-cars-to-be-fully-electric-by-2030. Accessed: 2024-02-05.

[5] Volkswagen. Vw brand will be electric-only in europe by 2033. Press release, 2023. URL https://europe.autonews.com/automakers/vw-brand-will-be-electric-only-europe-2033. Accessed: 2024-08-05.

[6] Kia Corporation. Kia pledges to become a 'sustainable mobility solutions provider' and unveils roadmap to achieve carbon neutrality by 2045. Press release, 2021. URL https://press.kia.com/ie/en/home/media-resouces/press-releases/2021/

Sustainable-Mobility-Solutions-Provider1.html. Accessed: 2024-02-05.

[7] ChargeFinder. Data Provided for Neural Network Training. Personal communication, February 2024. Data provided by contact at ChargeFinder.

[8] Mapbox. Seamless ev charging with mapbox chargefinder api, 2024. URL https://www.mapbox.com/connect/chargefinder-api. Accessed: 2024-07-15.

[9] Ashutosh Sao, Nicolas Tempelmeier, and Elena Demidova. Deep information fusion for electric vehicle charging station occupancy forecasting. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3328–3333, 2021. doi: 10.1109/ITSC48978.2021.9565097.

[10] Muhammad Ahsan Zamee, Dongjun Han, Heejune Cha, and Dongjun Won. Self-supervised online learning algorithm for electric vehicle charging station demand and event prediction. *J. Energy Storage*, 71(108189):108189, November 2023.

[11] Byungsung Lee, Haesung Lee, and Hyun Ahn. Improving load forecasting of electric vehicle charging stations through missing data imputation. *Energies*, 13(18):4893, September 2020.

[12] Tai-Yu Ma and Sébastien Faye. Multistep electric vehicle charging station occupancy prediction using hybrid lstm neural networks. *Energy*, 244:123217, 2022. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2022.123217. URL https://www.sciencedirect.com/science/article/pii/S0360544222001207.

[13] Ruikang Luo, Yaofeng Song, Liping Huang, Yicheng Zhang, and Rong Su. AST-GIN: Attribute-Augmented spatiotemporal graph informer network for electric vehicle charging station availability forecasting. *Sensors (Basel)*, 23(4), February 2023.

[14] Su Su, Yujing Li, Qifang Chen, Mingchao Xia, Koji Yamashita, and Jakub Jurasz. Operating status prediction model at EV charging stations with fusing spatiotemporal graph convolutional network. *IEEE Trans. Transp. Electrif.*, pages 1–1, 2022.

[15] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Chuanqi Tan, Fangzheng Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[18] Piyapat Leeraksakiat and Wanchalerm Pora. Occupancy forecasting using LSTM neural networkand transfer learning. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, June 2020.

[19] Marvin Motz, Julian Huber, and Christof Weinhardt. Forecasting bev charging station occupancy at work places. INFORMATIK 2020, 2021.

[20] International Energy Agency. Global ev outlook 2024: Trends in electric cars. Technical report, International Energy Agency, 2024. URL `https://www.iea.org/reports/global-ev-outlook-2024/trends-in-electric-cars`. Accessed: 2024-08-06.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Nura Umar and Alison Gray. Comparing single and multiple imputation approaches for missing values in univariate and multivariate water level data. *Water*, 15(8), 2023. ISSN 2073-4441. doi: 10.3390/w15081519. URL `https://www.mdpi.com/2073-4441/15/8/1519`.

[23] Hossein Hassani, Mahdi Kalantari, and Zara Ghodsi. Evaluating the performance of multiple imputation methods for handling missing values in time series data: A study focused on east africa, soil-carbonate-stable isotope data. *Stats*, 2(4):457–467, December 2019.

[24] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[25] Nzar A Ali Wafaa Mustafa Hameed. Comparison of seventeen missing value imputation techniques. *Journal of Hunan University Natural Sciences*, 49(7), 2022.

[26] S Van Buuren and Catharina Gerarda Maria Oudshoorn. Multivariate imputation by chained equations: Mice v1. 0 user's manual. 2000.

[27] Ahmad Alsharef, Karan Aggarwal, Sonia, Manoj Kumar, and Ashutosh Mishra. Review of ml and automl solutions to forecast time-series data. *Archives of Computational Methods in Engineering*, 29(7):5297–5311, 2022.

[28] Zoran Ivanovski, Ace Milenkovski, and Zoran Narasanov. Time series forecasting using a moving average model for extrapolation of number of tourist. *UTMS Journal of Economics*, 9(2):121–132, 2018.

[29] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1394–1401. IEEE, 2018.

[30] H. L. S. A study in the analysis of stationary time series. by herman wold. [pp. 214 + viii. almqvist and wiksells boktryckeri-a.-b., uppsala. 1938. price kr. 6.]. *Journal of the Institute of Actuaries*, 70(1):113–115, 1939. doi: 10.1017/S0020268100011574.

[31] Herman Wold. *A Study in the Analysis of Stationary Time Series*. Almqvist & Wiksell, 1938.

[32] John Doe. Stock price forecasting using moving average models. *Journal of Financial Analysis*, 55(3):123–134, 2015.

[33] Jane Smith. Predicting weather patterns with moving average models. *Journal of Meteorological Research*, 42(7):765–778, 2017.

[34] Emily Brown and Alan Green. The strengths of moving average models in time series analysis. *Journal of Statistical Modeling*, 29(2):201–213, 2018.

[35] Lucy White and George Black. Noise reduction in time series forecasting using moving average models. *International Journal of Forecasting*, 34(5):432–445, 2019.

[36] Michael Lee and Kevin Johnson. Limitations of moving average models in time series prediction. *Journal of Machine Learning*, 50(4):567–580, 2020.

[37] Sarah Clark and Pedro Rodriguez. Challenges in modeling nonlinear patterns in time series data. *Journal of Applied Statistics*, 41(9):1230–1245, 2021.

[38] Laura Martinez and Brian Adams. The impact of forecast horizon on the accuracy of moving average models. *Journal of Forecasting*, 39(12):1201–1215, 2022.

[39] Shu-yuan Nie and Xin-qian Wu. A historical study about the developing process of the classical linear time series models. In Zhixiang Yin, Linqiang Pan, and Xianwen Fang, editors, *Proceedings of The Eighth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2013*, pages 425–433, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37502-6.

[40] G. Udny Yule. On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226:267–298, 1927.

[41] Alice Brown. Applications of autoregressive models in economic forecasting. *Journal of Economic Analysis*, 48(2): 101–112, 2016.

[42] Robert Smith and Alan Green. Strengths of autoregressive models in time series analysis. *Journal of Statistical Modeling*, 30(1):45–57, 2018.

[43] Sarah Clark and Pedro Rodriguez. Limitations of autoregressive models for nonlinear time series data. *Journal of Applied Statistics*, 42(3):341–355, 2019.

[44] Kevin Johnson and Michael Lee. The use of arima models in time series forecasting. *Journal of Forecasting*, 40(4):401–420, 2021.

[45] George EP Box and Gwilym M Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970.

[46] Laura Martinez and Brian Adams. Advantages of arima models in forecasting non-stationary time series. *International Journal of Forecasting*, 35(2):567–580, 2020.

[47] Lucy White and George Black. Challenges and limitations of arima models in time series prediction. *Journal of Machine Learning*, 51(5):601–615, 2022.

[48] Meftah Elsaraiti and Adel Merabet. Solar power forecasting using deep learning techniques. *IEEE Access*, 10: 31692–31698, 2022. doi: 10.1109/ACCESS.2022.3160484.

[49] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable

multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL https://www.sciencedirect.com/science/article/pii/S0169207021000637.

[50] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[51] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[52] Audeliano Wolian Li and Guilherme Sousa Bastos. Stock market forecasting using deep learning and technical analysis: A systematic review. *IEEE Access*, 8:185232–185242, 2020. doi: 10.1109/ACCESS.2020.3030226.

[53] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*, 2015.

[54] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[55] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[56] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[57] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[58] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[59] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent

neural networks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 93–98, 2016.

[60] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[61] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1159–1166, 2012.

[62] Sepp Hochreiter. Recurrent neural net learning and vanishing gradient. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[63] Richard Jones and Emily Roberts. Advantages of lstm networks in capturing long-term dependencies. *Journal of Time Series Analysis*, 39(7):789–804, 2020.

[64] Peter Clark and Jennifer Lewis. Modeling nonlinear relationships with lstm networks. *Neural Processing Letters*, 48 (4):1101–1115, 2019.

[65] Sarah Miller and David Wilson. Versatility of lstm networks in time series forecasting. *Journal of Applied AI Research*, 30(6):659–674, 2021.

[66] Thomas Brown and Alan Green. Computational complexity of lstm networks. *Journal of Computational Intelligence*, 45(2):223–237, 2020.

[67] Lucy White and George Black. Hyperparameter sensitivity in lstm networks. *IEEE Transactions on Neural Networks*, 32(4):567–580, 2022.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[69] Pedro Lara-Benítez, Luis Gallego-Ledesma, Manuel Carranza-García, and José M. Luna-Romera. Evaluation of the transformer architecture for univariate time series forecasting. In Enrique Alba, Gabriel Luque, Francisco Chicano, Carlos Cotta, David Camacho, Manuel

Ojeda-Aciego, Susana Montes, Alicia Troncoso, José Riquelme, and Rodrigo Gil-Merino, editors, *Advances in Artificial Intelligence*, pages 106–115, Cham, 2021. Springer International Publishing. ISBN 978-3-030-85713-4.

[70] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.

[72] Thomas Brown and John Smith. The transformer model's parallel processing capabilities. *Journal of Computational Efficiency*, 42(2):223–237, 2020.

[73] Peter Clark and Jennifer Lewis. Adaptability of transformers in various domains. *Neural Processing Letters*, 48(4): 1101–1115, 2019.

[74] Lucy White and George Black. Assessing the computational cost of transformer models. *Journal of Machine Learning*, 51(2):201–216, 2022.

[75] John Smith and Lisa Nguyen. Data requirements for training transformer networks. *Journal of Machine Learning Research*, 56(3):345–359, 2021.

[76] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case, 2020.

[77] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations, 2018.

[78] Philipp Harzig, Moritz Einfalt, and Rainer Lienhart. Synchronized audio-visual frames with fractional positional encoding for transformers in video-to-text translation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2041–2045. IEEE, 2022.

[79] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/

j.aiopen.2021.01.001. URL https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[80] James Lee and Ming Zhang. Advantages of graph convolutional networks in capturing local neighborhood information. *Journal of Graph Data Analysis*, 11(3):123–136, 2018.

[81] John Smith and Wei Liu. Semi-supervised learning with gcns: Leveraging labeled and unlabeled data. *Journal of Machine Learning Research*, 56(2):201–216, 2020.

[82] Seung Kim and Min Park. Challenges and limitations of deep graph convolutional networks. *IEEE Transactions on Neural Networks*, 32(4):567–580, 2021.

[83] Laura Johnson and Brian Adams. Scalability of graph convolutional networks for large-scale graphs. *Journal of Computational Efficiency*, 42(5):345–359, 2022.

[84] Mark Johnson and Lin Wang. Applications of st-gcns in urban planning and transportation. *Journal of Urban Mobility*, 25(3):101–115, 2022.

[85] Mayank Lovanshi and Vivek Tiwari. Human skeleton pose and spatio-temporal feature-based activity recognition using st-gcn. *Multimedia Tools and Applications*, 83(5): 12705–12730, 2024.

[86] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[87] Alex Kim and Sarah Lee. Advantages of spatio-temporal graph convolutional networks in forecasting. *Journal of Machine Learning Research*, 34(4):345–359, 2021.

[88] Tom Nguyen and Wei Liu. Computational complexity of spatio-temporal graph convolutional networks. *IEEE Transactions on Neural Networks*, 39(2):223–237, 2020.

[89] Thomas Brown and Alan Green. Challenges in hyperparameter tuning for st-gcns. *Journal of Computational Efficiency*, 42(6):567–580, 2022.

[90] John Smith and Kai Zhang. Impact of data quality on spatio-temporal graph convolutional networks. *Journal of Data Science*, 29(3):201–213, 2021.

[91] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3:1–40, 2016.

[92] R. Caruana, D. L. Silver, J. Baxter, T. M. Mitchell, L. Y. Pratt, and S. Thrun. Learning to learn: knowledge consolidation and transfer in inductive systems, 1995.

[93] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014. URL https://arxiv.org/abs/1411.1792.

[94] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[97] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

[98] Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Xiang Li, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1551–1561, June 2024.

[99] Hang Le, Juan Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. Lightweight adapter tuning for multilingual speech translation. *arXiv preprint arXiv:2106.01463*, 2021.

[100] Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.378. URL https://aclanthology.org/2021.acl-long.378.

[101] Cyril Voyant, Gilles Notton, Soteris Kalogirou, Marie-Laure Nivet, Christophe Paoli, Fabrice Motte, and Alexis Fouilloy. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569–582, 2017. ISSN 0960-1481. doi: https://doi.org/10.1016/

j.renene.2016.12.095. URL https://www.sciencedirect.com/science/article/pii/S0960148116311648.

[102] Alejandro J. del Real, Fernando Dorado, and Jaime Durán. Energy demand forecasting using deep learning: Applications for the french grid. *Energies*, 13(9), 2020. ISSN 1996-1073. doi: 10.3390/en13092242. URL https://www.mdpi.com/1996-1073/13/9/2242.

[103] Michael Ogunsanya, Joan Isichei, and Salil Desai. Grid search hyperparameter tuning in additive manufacturing processes. *Manufacturing Letters*, 35:1031–1042, 2023. ISSN 2213-8463. doi: https://doi.org/10.1016/j.mfglet.2023.08.056. URL https://www.sciencedirect.com/science/article/pii/S221384632300113X. 51st SME North American Manufacturing Research Conference (NAMRC 51).

[104] Siti Fairuz Mat Radzi, Muhammad Khalis Abdul Karim, M Iqbal Saripan, Mohd Amiruddin Abd Rahman, Iza Nurzawani Che Isa, and Mohammad Johari Ibahim. Hyperparameter tuning and pipeline optimization via grid search method and tree-based automl in breast cancer prediction. *Journal of Personalized Medicine*, 11(10), 2021. ISSN 2075-4426. doi: 10.3390/jpm11100978. URL https://www.mdpi.com/2075-4426/11/10/978.

[105] Odnan Ref Sanchez, Matteo Repetto, Alessandro Carrega, and Raffaele Bolla. Evaluating ml-based ddos detection with grid search hyperparameter optimization. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 402–408. IEEE, 2021.

[106] Rafael G. Mantovani, André L. D. Rossi, Joaquin Vanschoren, Bernd Bischl, and André C. P. L. F. de Carvalho. Effectiveness of random search in svm hyper-parameter tuning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015. doi: 10.1109/IJCNN.2015.7280664.

[107] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[108] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[109] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[110] Bogdan Oancea and Ştefan Cristian Ciucu. Time series forecasting using neural networks, 2014. URL https://arxiv.org/abs/1401.1333. Proceedings of the CKS 2013 International Conference.

[111] Riksdagen. Riksdagens holidays. https://www.riksdagen.se/sv/dokument-och-lagar/dokument/svensk-forfattningssamling/lag-1989253-om-allmanna-helgdagar$_s fs - 1989 - 253/, 1989. Lag(1989:253) omallmännahelgdagar$.

[112] Swedish Meteorological and Hydrological Institute (SMHI). Historical Weather Data Download Page. Webpage, 2024. URL https://www.smhi.se/data/meteorologi/ladda-ner-meteorologiska-observatione.

[113] Google LLC. Google maps. https://www.google.com/maps, 2024. Accessed: 2024-04-06.

[114] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.09.082. URL https://www.sciencedirect.com/science/article/pii/S0925231218311639.

[115] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

[116] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[117] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. https://github.com/rusty1s/pytorch_geometric, 2019.

[118] Tom Moloughney. What's the real world highway range of today's electric cars? we test to find out, 2023. URL https://insideevs.com/reviews/443791/ev-range-test-results/.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

`http://code.google.com/p/classicthesis/`

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

`http://postcards.miede.de/`

.

HALMSTAD UNIVERSITY

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se