Forecasting Electric Vehicle Charging Station Occupancy and Route Recommendations through Deep Learning

Daniel Persson William Wahlberg

April 10, 2024

Abstract

The widespread adoption of electric vehicles (EVs) has increased immensely during these last years. As the number of EVs on the roads continues to increase, the demand for efficient and reliable charging infrastructure becomes increasingly important. In this context, forecasting charging station occupancy is a critical element in the effective management of EV charging infrastructure.

This project aims to develop a method for predicting the occupancy of electric vehicle (EV) charging stations and to provide route recommendations for EV drivers using deep learning techniques. The prediction aspect will leverage a Spatio-Temporal Graph Neural Network (ST-GCN) model built on top of a Transformer network. This network will be adept at capturing complex spatial and temporal relationships in data. For the route recommendation, a Deep Reinforcement Learning (DRL) agent will be employed. This agent will utilize the predictions of charging station occupancy to plan the most efficient route for EV drivers. The neural networks will be trained on data provided by ChargeFinder [1], a platform that offers real-time information on the availability and usage patterns of charging stations. By using deep learning, this project aims to extract intricate patterns from historical data to enable more accurate forecasts of future charging station occupancy. The ultimate goal of this project is to contribute to the enhancement of EV charging infrastructure efficiency and responsiveness.

Contents

1 Introduction **3**
- 1.1 Problem Definition . . . 3
- 1.2 Research questions . . . 4
- 1.3 Goal . . . 4
- 1.4 Data . . . 4

2 Related works **4**
- 2.1 Data imputations . . . 4
- 2.2 Algorithms . . . 5
  - 2.2.1 MA . . . 5
  - 2.2.2 AR . . . 5
  - 2.2.3 ARIMA . . . 6
  - 2.2.4 LSTM . . . 6
  - 2.2.5 Transformer . . . 7
  - 2.2.6 GCN . . . 8
  - 2.2.7 ST-GCN . . . 8
- 2.3 DRL Agent . . . 9
- 2.4 Analysis And Optimization . . . 9
  - 2.4.1 Regression Evaluation . . . 9
  - 2.4.2 DRL Agent Evaluation . . . 10
  - 2.4.3 Hyperparameter Optimization . . . 10

3 Project plan **10**
- 3.1 Agile . . . 11
- 3.2 Expected Novelty . . . 12

4 Method **12**
- 4.1 Data description . . . 12
- 4.2 Preprocessing . . . 12
- 4.3 Routes . . . 14
- 4.4 Horizontal Line . . . 16
- 4.5 ML Baseline Methods . . . 16
- 4.6 DL Baseline Methods . . . 19
  - 4.6.1 LSTM . . . 19
  - 4.6.2 Transformers . . . 19
- 4.7 ST-GCN . . . 21

5 Results **22**
- 5.1 Routes . . . 22
- 5.2 MA . . . 24
- 5.3 AR . . . 24
- 5.4 ARIMA . . . 24
- 5.5 LSTM . . . 24
- 5.6 Transformer . . . 25
  - 5.6.1 Encoder-only model . . . 25
  - 5.6.2 Encoder-Decoder . . . 26
- 5.7 Our ST-GCN . . . 26
- 5.8 Agent . . . 26

6 Conclusion **29**

1 Introduction

The EV industry is undergoing a considerable growth on a global scale. In 2022, sales exceeded 10 million, and the proportion of EVs in total sales tripled from approximately 4% in 2020 to 12% in 2022 [2]. This growth indicates that the transport sector is shifting from fossil-powered to electric-powered vehicles.

In Europe, the momentum towards electrification is mirrored in the commitments of major car manufacturers. Volvo, Volkswagen, and KIA have pledged to transition to fully electric models by 2030, 2033, and 2035, respectively [3][4][5]. Aligned with this transformative shift, the European Green Deal stands as a pivotal initiative aiming to lead the continent towards climate neutrality. Encompassing a vision of a forward-thinking, resource-efficient, and competitive economy, the deal pledges to achieve net-zero greenhouse gas emissions by 2050, and it actively promotes the decoupling of economic growth from resource consumption while championing inclusivity for all [6]. This commitment is further solidified by the recent accord between the European Parliament and Council, declaring that all new cars and vans within Europe will be emission-free by 2035. As part of this transition, updated CO2 standards mandate a 55% reduction in new car emissions and a 50% reduction for new vans by 2030 [7].

In Sweden, a country where the impact of these changes is particularly pronounced due to the prevalence of major car manufacturers like Volvo, Volkswagen, and KIA on local roads, the commitment to full electrification is unmistakable. These manufacturers, although not exclusive to Sweden, are exceptionally common on Swedish streets. Their goal to transition to fully electric models resonates deeply within the Swedish automotive landscape. This underscores Sweden's role in embracing sustainable mobility practices, aligning with broader European objectives and reinforcing its commitment to combat climate change. The narrative in Sweden reflects a proactive stance, exemplifying a collective effort toward environmentally conscious transportation within the context of the European Green Deal.

However, while the number of electric cars is increasing on the road, so is the load on the charging infrastructure. Forecasting the occupancy (predicting how many chargers at a charging station are occupied) of EV charging stations is crucial for optimizing usage and improving user experience. It aids city planners and companies in effectively developing charging infrastructure and it is also a great convenience to EV drivers. Having access to reliable information and predictions about the availability of free chargers could encourage drivers to utilize multiple charging stations in a region rather than clustering around a nearby one.

1.1 Problem Definition

Since the number of EVs sold globally is increasing, as mentioned earlier, the demand for efficient charging infrastructure is intensifying. This research problem is particularly relevant for users reliant on EVs for daily commuting or travel, emphasizing the need for accurate charging station occupancy predictions to enhance user experience and optimize infrastructure efficiently. Researchers have approached this problem in two primary ways:

- Single Charging Station Forecasting. Some studies use time series forecasting methods to predict how often individual charging stations will be occupied. This involves analyzing past usage patterns to estimate future availability[8][9][10][11].
- Multiple Charging Station Forecasting. Other studies expand this approach to include several charging stations at once[12][13]. This broader perspective helps to understand the availability of charging stations across a wider area, not just at a single location.For example, when two stations are nearby and one consistently fills up before the other, it becomes possible to predict an increase in usage at the second station as the first approaches full capacity.

However, these methods have a limitation: they don't necessarily guide users to the best charging locations. For instance, it might seem better to head for a single charger with a 50% chance of being available, but it could actually be more advantageous to go to a location with three chargers, each with a 30% chance of availability. This is because the probability of finding at least one free charger among the three is higher (around 66%).

The central challenge addressed in this thesis is to offer intelligent route recommendations to electric vehicle (EV) drivers, leveraging occupancy forecasts as a key factor in the decision-making process. It's not just about predicting which charging stations will be occupied but also about guiding the users to the best possible location for charging. This involves considering the number of chargers in an area and their individual chances of being free, and then making a smart decision about the most suitable charging location based on charger availability and proximity.

1.2 Research questions

- Does the dataset from ChargeFinder demonstrate a level of occupancy pattern predictability that significantly surpasses what would be expected by chance alone?
- How does the application of transfer learning techniques impact the analysis of ChargeFinder data, particularly in terms of enhancing predictive performance and reducing training time across comparable city datasets?.
- Can the DRL agent be extended for energy efficiency?

1.3 Goal

The first objective of this thesis is to employ an ST-GCN model to forecast the occupancy levels of multiple individual charging stations while also integrating various external data sources, such as weather conditions and holidays. ST-GCNs are specialized neural networks designed to analyze and model both spatial and temporal patterns within the data. This is achieved by integrating graph convolutional operations with recurrent layers. By simultaneously predicting the occupancy of multiple charging stations, the predictions are not solely reliant on the characteristics of a single station itself; instead, they take into account the connections and interactions with neighboring stations.

The second objective of this thesis is to develop a smart path recommendation system for pinpointing the best charging locations. The path recommendation system will employ a DRL agent. This agent is designed for more than just choosing available chargers; it strategically assesses the proximity of the locations to these chargers, thereby improving the usefulness of its recommendations. The agent will be built upon the outputs of the ST-GCN, utilizing its forecast data as training inputs alongside the graph modeled after the road network. This integration ensures that the agent's recommendations are both data-driven and practically oriented for optimal charger location selection.

1.4 Data

This thesis leverages data from ChargeFinder [1]. The dataset is characterized by an intermittent nature. Unlike structured datasets with continuous intervals, intermittent datasets record data points only during user interactions with the ChargeFinder website, presenting a sporadic distribution. Figure 1 visually illustrates the intermittent nature of the *created_at* feature, showcasing the separation of the data across four distinct groups over four consecutive days. Notably, data is almost absent during nighttime periods due to the minimal querying activity on the website, leading to gaps in the dataset.

2 Related works

In the upcoming sections, several related works are examined, emphasizing the application of algorithms, data imputation techniques, and analytical methods to showcase innovative strategies that could prove beneficial for this project.

2.1 Data imputations

To convert intermittent time series data, as shown in Figure 1, into a continuous format, it is necessary to utilize imputation techniques. Imputation methods are typically classified into two categories: univariate and multivariate. Univariate imputation involves filling missing values within a single variable using only information from that variable, while multivariate imputation considers relationships and dependencies between multiple variables to achieve more accurate missing data imputations.

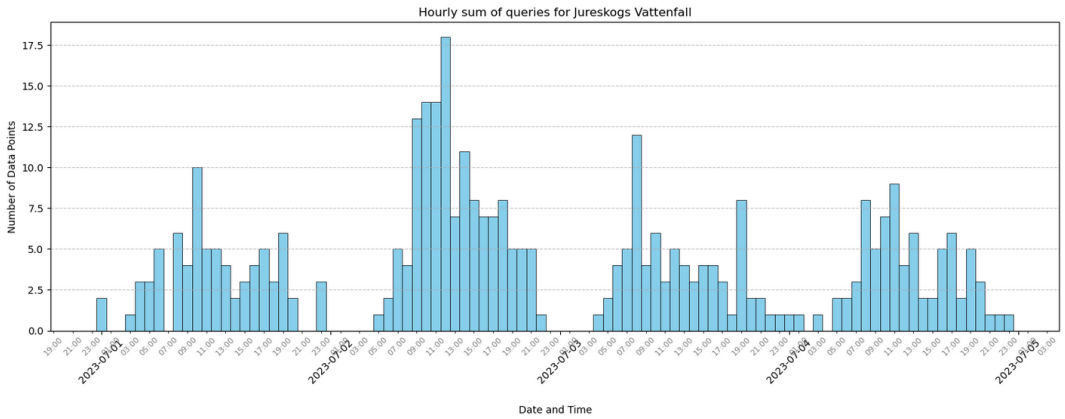

Figure 1: An extract consisting of four days from the jureskogs_vattenfall dataset and their hourly frequency of data-points.

In a project with a similar intermittent dataset [14] to the one used in this thesis, temporally-based univariate strategies are employed. These strategies include two methods: Last Observation Carried Forward (LOCF) and Next Observation Carried Backward (NOCB). LOCF entails carrying the last observed value before a missing data point forward to substitute for the missing value, while NOCB involves carrying the next observed value after a missing data point backward to substitute for the missing value. Both strategies operate under the assumption that the preceding measurement (for LOCF) or the succeeding measurement (for NOCB) serves as a reasonable estimate. In addition to these temporal strategies, interpolation data imputation is a technique that estimates missing or incomplete data points within a dataset by filling in values based on existing data, utilizing a mathematical method. Within the realm of multivariate imputation, methodologies such as the k-Nearest Neighbors (k-NN) technique are explored in [15]. k-NN fills missing values in a dataset by considering the values of their nearest neighbors. This technique leverages the similarity between data points to estimate and replace missing information.

2.2 Algorithms

The following sections will explore some methodologies that are commonly used for time-series forecasting. Moving Average(MA), AutoRegressive(AR), and AutoRegressive Integrated Moving Average(ARIMA) are simple linear Machine Learning (ML) models that are used for time series prediction[16][17][18]. Long short-term memory(LSTM), Transformer, graph convectional network (GCN), and ST-GCNs are more advanced DL models that can also be implemented for time series prediction.

2.2.1 MA

An MA model aims 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)[18], as shown in Equation 1:

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{1}$$

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 $t - i$.

2.2.2 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[18], as shown in Equation 2.

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t \tag{2}$$

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. ε_t represents the error term at time t.

2.2.3 ARIMA

$ARIMA$, denoted as $ARIMA(p, d, q)$, combines AR, differencing (I), and MA components to model time series data, as shown in Equation 3.

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + t + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{3}$$

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. 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. ε_{t-i} represents the error term at time $t - i$.

2.2.4 LSTM

In the realm of time series forecasting, due to the limitations inherent in traditional machine learning models—such as their inability to capture long-term dependencies, handle high-dimensional data effectively, and model complex nonlinear relationships—researchers have increasingly shifted their focus towards deep learning methods over the past decade. LSTM networks, in particular, have played a crucial role due to their effectiveness in handling time series data. 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[19][11][20]. Originating from the seminal paper [21], LSTM networks have been extensively adopted across various domains. For instance, [22] 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.

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. Please see Equations 4 to 9, fetched from this article [23].

$$f_t = \sigma(X_t U^f + S_{t-1} W^f + b_f) \tag{4}$$

$$i_t = \sigma(X_t U^i + S_{t-1} W^i + b_i) \tag{5}$$

$$\tilde{C}_t = \tanh(X_t U^c + S_{t-1} W^c + b_c) \tag{6}$$

$$C_t = C_{t-1} \odot f_t + i_t \odot \tilde{C}_t \tag{7}$$

$$o_t = \sigma(X_t U^o + S_{t-1} W^o + b_o) \tag{8}$$

$$S_t = o_t \odot \tanh(C_t) \tag{9}$$

Here, $f_t = \sigma(X_t U^f + S_{t-1} W^f + b_f)$ (Equation 4) is the forget gate that controls the information flow from the previous cell state C_{t-1}; $i_t = \sigma(X_t U^i + S_{t-1} W^i + b_i)$ (Equation 5) is the input gate that controls the information flow to the current cell state C_t; $\tilde{C}_t = \tanh(X_t U^c + S_{t-1} W^c + b_c)$ (Equation 6) is the candidate cell state that contains new candidate values; $C_t = C_{t-1} \odot f_t + i_t \odot \tilde{C}_t$ (Equation 7) is the updated cell state that combines the previous cell state with the candidate values; $o_t = \sigma(X_t U^o + S_{t-1} W^o + b_o)$ (Equation 8) is the output gate that controls the information flow from the current cell state C_t. $S_t = o_t \odot \tanh(C_t)$ (Equation 9) is the hidden state output that contains the filtered information based on the output gate.

2.2.5 Transformer

In 2017, a groundbreaking paper introduced the Transformer neural network architecture [24], which demonstrated exceptional performance in natural language processing tasks. Given the similarities between time series prediction and natural language modeling, the Transformer architecture shows good results in the realm of time series forecasting as well. Transformers work with sequence-to-sequence generation by using an Encoder-Decoder architecture. The encoder processes input data to extract meaningful representations, while the decoder generates output sequences based on these representations and on the previous predictions. Distinct from the recurrent-based approach of LSTMs, the Transformer architecture, both in its encoder and decoder components, exclusively utilizes attention mechanisms to process input data. This allows the model to dynamically focus on relevant parts of the input when making predictions, significantly improving its ability to understand context and generate accurate predictions. Moreover, the Transformer incorporates a multi-headed attention approach, enabling it to process various aspects of the information in parallel, please see Equations 10-11. This self-attention process handles all items in a sequence concurrently, offering a different approach to sequential data analysis.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

Where Q Represents the matrix of queries, K stands for the matrix of keys, 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.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{11}$$

Where QW_i^Q, KW_i^K, VW_i^V are the weight matrices for the queries, keys, and values, respectively, head_1 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.

The input embedding layers in the Transformer architecture, as introduced in [24], serve to convert tokens into numerical vectors that can be processed by the model. This is essential in NLP tasks, where the model needs to interpret text data. However, for projects where the data is already in numerical format, the necessity for embedding layers to perform token-to-vector transformations diminishes. In such scenarios, it's common practice to bypass the standard embedding layers. Instead, a linear layer is employed to adjust the input data to the appropriate size for the encoder or decoder inputs[25][26].

In the realm of transformer models, particularly when applied to time series data, positional encoding plays a pivotal role in conveying the sequence order to the model. A prominent approach is the Absolute Positional Encoding, as detailed [24][26][27]. This method employs specific equations, i.e., 12 and 13, to integrate the absolute positional information of tokens into the model, allowing it to perceive the sequence order.

For a position pos and a dimension i within the encoding vector, the sine and cosine functions are defined as:

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

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

Where PE represents the positional encoding, pos denotes the position within the sequence, i is the dimension, and d_{model} is the dimensionality of the embedding.

In contrast, Relative Positional Encoding [28] shifts the emphasis from the absolute positions to the relative distances between tokens. This approach does not encode the absolute position of a token; rather, it focuses on the relative positioning between tokens. 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 then learns embeddings for these relative positions, enabling it to understand and leverage the positional relationships between tokens in a sequence more effectively. This method offers a nuanced perspective on how tokens influence each other based on their relative distances, enhancing the model's ability to generalize across different sequence lengths and structures.

Another crucial consideration in adapting Transformer models from NLP to time series analysis is determining whether the task at hand is a classification or a regression problem. This distinction

significantly impacts how one might modify the original Transformer architecture to suit the specifics of the time series task. For regression problems, it's essential to address the structure of the decoder's output head, which, in its default configuration for NLP tasks, typically includes a softmax layer designed for classification tasks. In the context of a regression problem, the softmax layer is not appropriate due to the continuous nature of the output. A common approach to adapt the model for regression is to remove the softmax layer[27][29], allowing the final linear layer's outputs to serve as the predictions. This adjustment simplifies the model's output mechanism to directly produce continuous values, aligning with the regression task's requirements.

During the training phase of Transformer models, a particular learning rate schedule is proposed in the seminal paper[24]. This schedule, detailed in Equation 14, is designed to optimize the training process by adjusting the learning rate accordingly to the number of 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:

$$\text{lrate} = d_{\text{model}}^{-0.5} \cdot \min\left(\text{step\_num}^{-0.5}, \text{step\_num} \cdot \text{warmup\_steps}^{-1.5}\right) \tag{14}$$

Here, $lrate$ represents the learning rate, d_{model}denotes the model's dimensionality, $step_num$ is the current step number, and $warmup_steps$ 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[24].

In the context of both time series forecasting and NLP, Transformer models are designed to predict future values or tokens based on the information available up to a certain point. A fundamental challenge in this endeavor is ensuring that the model does not inadvertently "peek" into the future data it is supposed to predict. This is addressed through two critical techniques: shifting the decoder input to the right, and applying causal masks. These strategies are widely discussed and utilized across various studies, including those focused on positional encoding and causality in time series forecasting[26][30][31][32], as well as in the original paper[24]. The shifting of the decoder input to the right ensures that at any point during training, the decoder has only access to previously seen tokens or data points, thus simulating the prediction scenario during inference where future data points are unknown. Causal masks complement this by preventing the self-attention mechanism in the decoder from attending to future positions. This is achieved by setting the attention weights to zero for positions that correspond to future data points, effectively making them invisible to the model during the computation of attention scores.

2.2.6 GCN

To perform computations on elements within a spatial context, GCNs can be utilized[33]. GCNs are an adaptation of Convolutional Neural Networks (CNNs). However, 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.

The original paper where it was introduced in 2016[33], the authors demonstrate 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 these semi-supervised settings, showcasing its potential for a wide range of applications in graph-structured data analysis.

2.2.7 ST-GCN

In some applications where forecasting is done on data with spatial elements, extra operations can be done to maximize the model's results. One example of such a network is an ST-GCN. Particularly interesting for this thesis is the work in [13], where ST-GCN is used to forecast occupancy at charging stations, proposing a Gated Recurrent Unit (GRU) to capture the temporal aspect of the data and a GCN for the spatial aspect. First, a graph containing the time series data is processed by the GCN, then, the GRU uses the time series produced by the GCN to forecast a prediction. The graph used

in this paper[13] is fully modeled as a city road network. Meaning that the graph not only includes the charging stations as nodes, but also includes intersections between the stations as nodes. Another similar paper [12] employed an ST-GCN by combining the GCN with an Informer instead of a GRU. While GRU processes data sequentially, an Informer network can process all parts of the sequence simultaneously. Unlike the work in [13] that fully modeled a graph after a road network, [12] took a shortcut by only considering the charging stations and the distance between them.

2.3 DRL Agent

The smart charging recommendation agent will be developed using DRL, a technique that combines the pattern recognition abilities of DL with the trial-and-error decision-making process of reinforcement learning. This method allows the agent to learn and make decisions by interacting with its environment, gaining insights from feedback through rewards or penalties, and recognizing patterns in complex data using neural networks similar to the human brain. DRL has also previously been used for smart path planning, a field similar to this project's application. In [34], the authors demonstrate that GCNs, inherently designed to understand graphs, could enable RL agents to learn routing optimization in a digital network. The same principle could work well in the application of route planning for a car. However, instead of using switches, access points, computers, and cables, as nodes and edges, intersections, charging stations, and roads will be conceptualized as nodes and edges. In [34], the DL agent employed deep Q-learning[35], defining its action space as potential network paths and measuring rewards based on route efficiency. Other DRL techniques, like policy-based methods and temporal difference learning, are also noteworthy, as detailed in [36]. Another similar project involves developing an Electric Vehicle Navigation System (EVNS) using DRL. It optimizes the on-the-go EV charging route and station selection by considering dynamic traffic conditions and charging requests [37]. However the difference between their project and this project is that [37] implements a model-free solution for greater flexibility and scalability. This project's DRL agent will build upon the model and occupancy from the ST-GCN.

One solution similar to the route recommendation method proposed in this project is the DRL method developed in [38]. This approach attempts to solve the problem by considering it as a scheduling problem and using a Markov Decision Process (MDP), focusing on minimizing EVs' total charging time and additional travel distance. The core of this approach is the deep Q-learning, which combines Q-learning with deep neural networks to learn complex patterns in charging station usage and EV behaviors. The algorithm effectively balances between slow and fast charging options, considering the limited travel distance of EVs due to battery constraints. The real-world implementation of this DRL approach shows a reduction in charging times compared to traditional methods. In particular, fine-tuning the system's parameters enables prioritization between minimizing total charging duration or minimizing additional travel distance, making it a versatile solution for urban EV charging needs.

2.4 Analysis And Optimization

The following sections will focus on the evaluation of the model results and the DRL agent's performance.

2.4.1 Regression Evaluation

Effective analysis of prediction results is crucial for refining forecasting models, thereby enhancing their accuracy and efficiency. Evaluation metrics like Mean Absolute Percentage Error (MAPE), Mean Bias Error (MBE), and Mean Square Error (MSE) are commonly employed in regression problems for comprehensive insights into prediction performance [39][40][19]. MAPE evaluates percentage accuracy, while MSE quantifies error magnitude. The equations for these three metrics are as follows:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100 \tag{15}$$

$$\text{MBE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i) \tag{16}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{17}$$

Where n is the total number of observations or data points in your dataset,Y_i represents the label for the i^{th} observation, $\hat{Y}_i$ represents the predicted value for the i^{th} observation.

2.4.2 DRL Agent Evaluation

In this project, the Deep Reinforcement Learning (DRL) agent's effectiveness is benchmarked against a basic strategy called Nearest Neighbor Charging Routing (NNCR), as detailed in prior research [41]. The NNCR method, which aims for simplicity, directs electric vehicles (EVs) to the nearest charging station without considering the availability of charging ports, potentially overlooking scenarios where the nearest station is fully occupied.

The performance metric used to compare the DRL agent and NNCR is the time taken to complete a journey, including the time spent traveling to the station and the potential waiting time if no immediate charging port is available. While NNCR minimizes the travel distance, it may not always minimize the total time, especially in situations where EVs are directed to occupied charging stations, leading to increased waiting times.By contrast, the DRL agent is expected to account for a broader set of factors, potentially offering more optimized routing and charging solutions by evaluating future outcomes and a wider array of variables.

2.4.3 Hyperparameter Optimization

There are many methods to optimize a model's hyperparameters, and one of the simplest approaches is using grid search. Grid search is an exhaustive optimization algorithm that methodically explores a wide range of parameter combinations, simultaneously applying cross-validation to each to determine the most effective set of parameters for a given model. [42][43][44].

In addition to grid search, another popular method for hyperparameter optimization is random search. Unlike grid search, which systematically evaluates all possible combinations, random search selects random combinations of parameters to evaluate, providing a more efficient approach when dealing with a large search space. This method can often find a good set of parameters quicker than grid search because it does not require the exhaustive evaluation of all combinations, thereby reducing the computational expense. [45][46]

3 Project plan

The initial phase of the project involves a comprehensive preprocessing strategy to enhance the quality and richness of the dataset. In this stage, NOCB and LOCF methods will be implemented, along with rounding interpolation, to compare these approaches in the final stage. The preprocessing stage will also integrate the k-NN algorithm to deal with missing values. In addition to imputation techniques, additional datasets will be integrated during preprocessing to augment the ChargeFinder dataset. This integration includes weather information sourced from SMHI [47] and official holiday data from the Parliament of Sweden[48]. The aim is to complement the existing dataset, providing a more comprehensive foundation for subsequent analyses.

Following this, diverse baseline ML models such as MA, AR, and ARIMA as well as DL baselines, including LSTM and Transformer networks, will be implemented and tested on the pre-processed dataset. These methods are expected to provide relatively good predictions, serving as benchmarks.

After implementing these baseline methods, the project proceeds with the development of a graph representation, an essential aspect of the spatial-temporal modeling technique. This graph is designed to model the spatial connections among all charging stations and it will include nodes representing intersections and charging stations, with edges indicating distances between these points. The distances are manually determined using Google Maps[49]. Once this graph is created, the approach involves implementing an ST-GCN model that combines a transformer-based network for temporal aspects and a GCN for spatial aspects. This unique approach should enhance the accuracy of EV charging station occupancy forecasts by utilizing the Transformer's ability to capture sequential data patterns and the GCN's proficiency in handling spatial relationships, similar to the road-based network in this project.

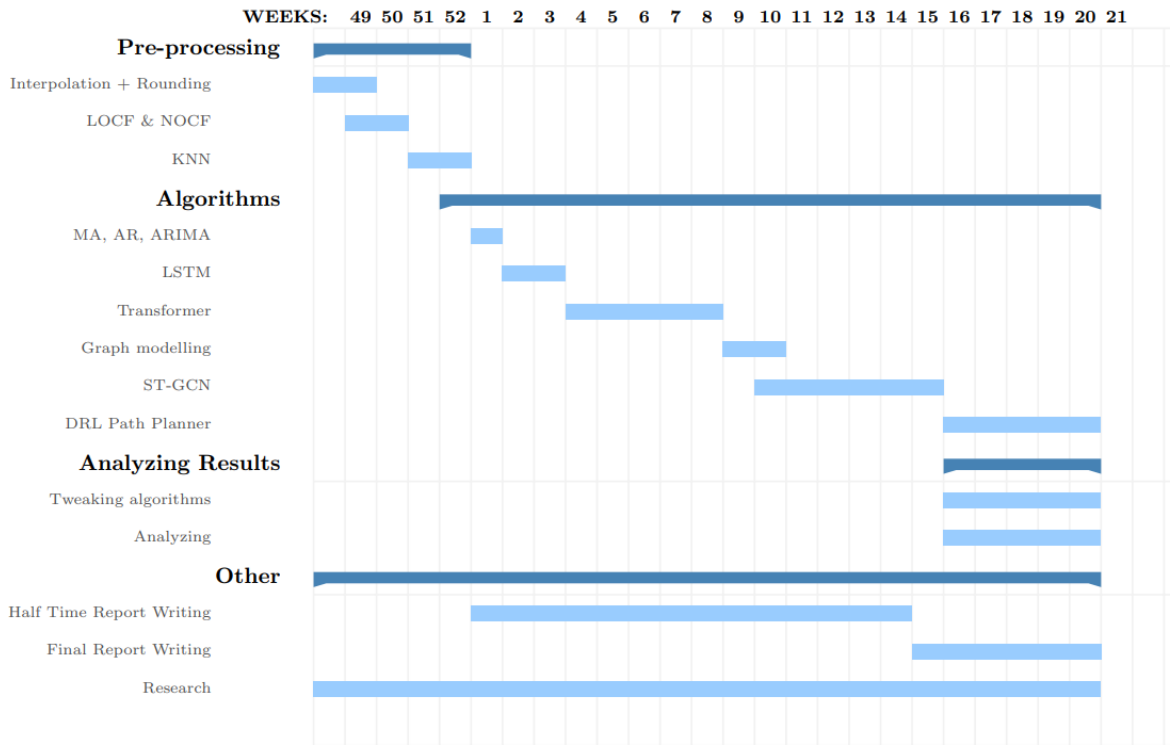

Figure 2: The project plan for this thesis.

The resulting network architecture aligns with the structure depicted in Figure 18, where *Chargefinder data* is represented as a graph structure with charging stations and intersections as nodes and roads as edges, subsequently utilized for GCN training.

The output obtained from the trained GCN is then integrated with *Auxiliary data*, which includes information such as public holiday schedules and weather conditions. It is important to notice that the preceding explanation pertains to a specific timestamp t. Furthermore, this process is performed for a range of timestamps from time t-n to t.

All the information from these timestamps is then fed into a Transformer neural network, which is responsible for generating predictions regarding the occupancy levels of all stations within the graph.

In the final phases of the project, a DRL agent will be trained to strategically determine the optimal charging location along a specific route. Unlike a simplistic approach of selecting available chargers, this agent aims to identify the best driving area based on proximity to chargers. Crucially, the reinforcement learning model will be trained using predictions from the ST-GCN.

3.1 Agile

The execution of this project will adhere to the principles of the Agile methodology. Agile represents an iterative approach to software development, emphasizing adaptability and continuous feedback. In accordance with Agile practices, the project will be segmented into diverse sub-goals, each assigned its own dedicated time-frame. This structured division of tasks is reflected in the Gantt schedule outlined in Figure 2.

Meetings will be held on a weekly basis with the project supervisors to give feedback on project progress and direction. In accordance with the Agile methodology, the scope of this project will remain flexible. For instance, should another ML method prove more suitable for the project than the ones currently used, and if the timeline permits, it will be incorporated into the project.

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 1: Data columns information for the IONITY dataset.

3.2 Expected Novelty

This research introduces advancements in predicting the occupancy of EV charging stations. The focus of this project is the integration of an ST-GCN with a Transformer network. The network will then be applied to the dataset provided by ChargeFinder. Prior to this project, the ChargeFinder dataset has not undergone any rigorous multistep prediction analysis.

4 Method

The following sections delve into the details of the datasets, discussing their seasonality, stations connections with each other, and unique characteristics such as the intermittent nature. Subsequently, the pre-processing methodologies employed to refine the raw data for effective modeling are elucidated. Additionally, the methodology for the baseline models as well as the ST-GCN is explained. However, any description concerning the DRL agent is missing due to there being no progress. The DRL agent will be implimented after the ST-GCN.

4.1 Data description

The datasets used in this project consist of data from multiple charging stations, each represented by seven columns that capture various attributes related to EV charging. These attributes are outlined in Table 1. The column *station_id*, provides a unique identifier for each station. This identifier can be cross-referenced with a lookup sheet to obtain the real station name and location. Additionally, the dataset contains columns such as *outlet_count*, indicating the number of charging outlets at each station, as well as *unknown_count*, *available_count*, *occupied_count*, and *offline_count*, which represent the counts of charging stations in different states: unknown, available, occupied, and offline, respectively. These counts are crucial for understanding the current status and utilization of the charging infrastructure. Finally, the *created_at* column, holding timestamp information, is of object type and provides records (in the form of date and time) about when a user queries the ChargeFinder website.

Thanks to its comprehensive features, this dataset offers an overview of the charging station landscape, enabling detailed analysis and prediction of occupancy patterns for enhanced operational efficiency. An extract of the dataset corresponding to the charging station near McDonald's in Värnamo can be seen in Figure 3.

4.2 Preprocessing

During the data pre-processing phase, the datasets undergo a transformation to augment their relevance and effectiveness in predicting EV charging station occupancy. First, a DateTime object is generated from the `created_at` column to capture the timestamp of each observation. Subsequently, more features are added to the data: `day_of_week` and `month` are included to facilitate temporal analysis, while `is_weekend`, `hour_bin`, and `is_holiday` are introduced to provide insights into weekend occurrences, time-of-day categorization, and holiday identification.

Notably, the *is_holiday* feature is created by considering public holidays in Sweden [48] throughout the period, including December 25th to January 1st, recognizing that many people are on holiday during this time. Figure 4 depicts an illustrative extract of the processed dataset. Additionally, a weather extract from the Swedish Meteorological and Hydrological Institute (SMHI) containing several

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

DateTime	station_id	outlet_count	unknown_count	available_count	occupied_count	offline_count	day_of_week	month	is_weekend	hour_bin	is_holiday
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 4: An extract of the processed dataset.

different weather features is incorporated into the dataset. A correlation matrix is then executed to determine if there are any significant correlations between the diverse 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, because in colder conditions, there could be higher demand as batteries tend to drain faster. Features with high correlation coefficients are retained for further modeling, as they are likely to provide valuable information for predicting occupancy. On the contrary, features with low correlation coefficients will be discarded.

Furthermore, when plotting the stations' number of chargers over time, it becomes evident that this quantity changes over time (see figure 5). To ensure consistency in our analysis, we normalized the occupancy data by recalculating the occupancy as *occupancy_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.

To gain a visual understanding of potential trends within the dataset, a comprehensive exploration is conducted by plotting the daily averages across the entire temporal span. An example of the IONITY charging station can be seen in Figure 6. Across all datasets, similar spikes are observed on certain dates, notably around Christmas, Easter, and an overall increase in July and August. Additionally, a

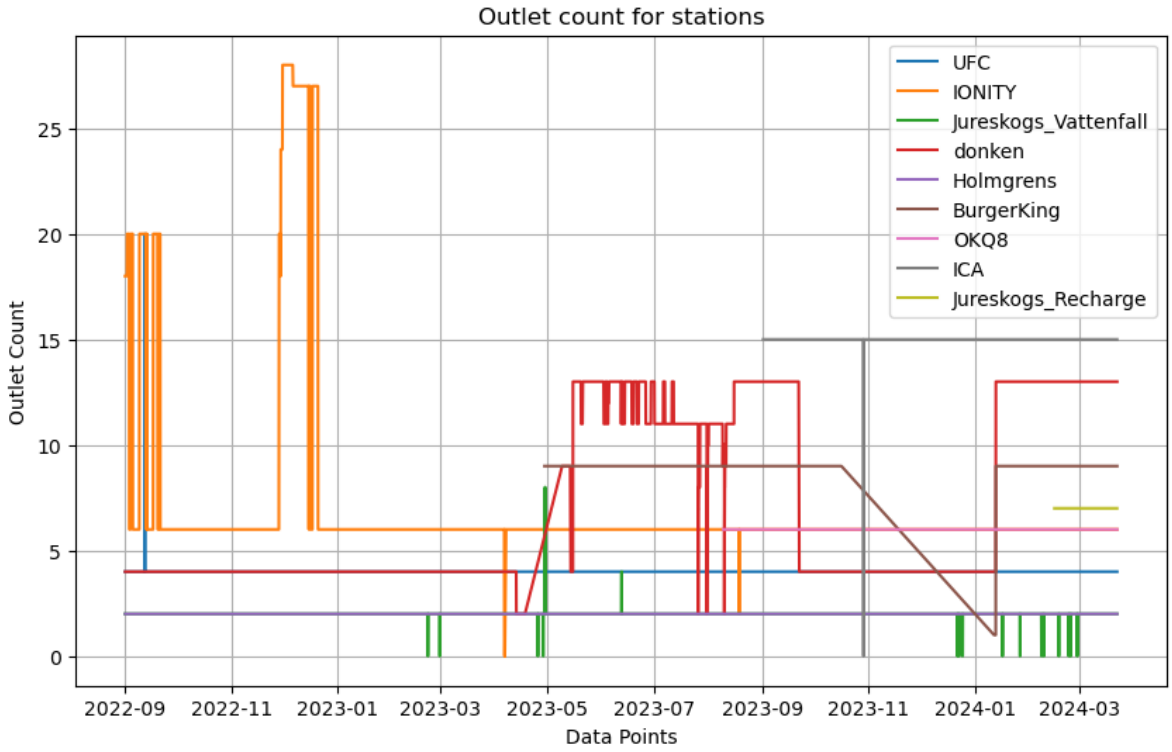

Figure 5: The total number of chargers/station over time.

significant drop in occupancy is observed during September and early December in 2022. These drops coincide with a sudden and temporary increase in the number of chargers, as shown in Figure 5. This discrepancy leads us to assume that the data is faulty, and that the total number of chargers during this time was actually 6. This would explain the temporary decrease during this period in the IONITY dataset, as the *outlet_count* is divided by a higher number of chargers than there actually are. One potential solution to fix this is to cap outlet_count to six for the IONITY charging station.

When examining the *created_at* column, notable gaps in data were detected, especially between 22:00 and 06:00. These gaps are clearly visible in Figure 1 and are consistent for every night in the dataset. Data imputation was used in this project to convert the dataset from an intermittent dataset into a continuous one. A sample day with all imputation methods can be seen in Figure 7. k-NN was chosen as the most appropriate method for this project because it produces a smoother time series that is easier to make predictions on. In contrast, having data that is more spiky data will mean that it will be more difficult for the algorithm to learn.

For a more comprehensive exploration and in-depth analysis of the temporal dynamics of EV charging station occupancy, a plot of the 5 different charging stations utilized are generated using time series decomposition. This analysis, as depicted in Figure 8, is performed using the *scikit-learn* library [50]. The trend is performed on daily averages, focusing on the *occupied_count* of the charging stations, depicting the long-term behavior, seasonalty, and overall directions. The detailed analysis of the trend decomposition across the five charging stations reveals pronounced seasonal patterns and trends, highlighting a consistent recurrence of peaks during specific periods. These peaks are mainly observed 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 stations' dynamic occupancy rates, which not only manifest a clear seasonal rhythm but also point to periods of heightened demand.

4.3 Routes

The depicted route showcased in Figure 9, spanning from Helsingborg to Jönköping, Sweden, has been purposefully chosen due to its substantial demand. This route contains a total of 54 strategically

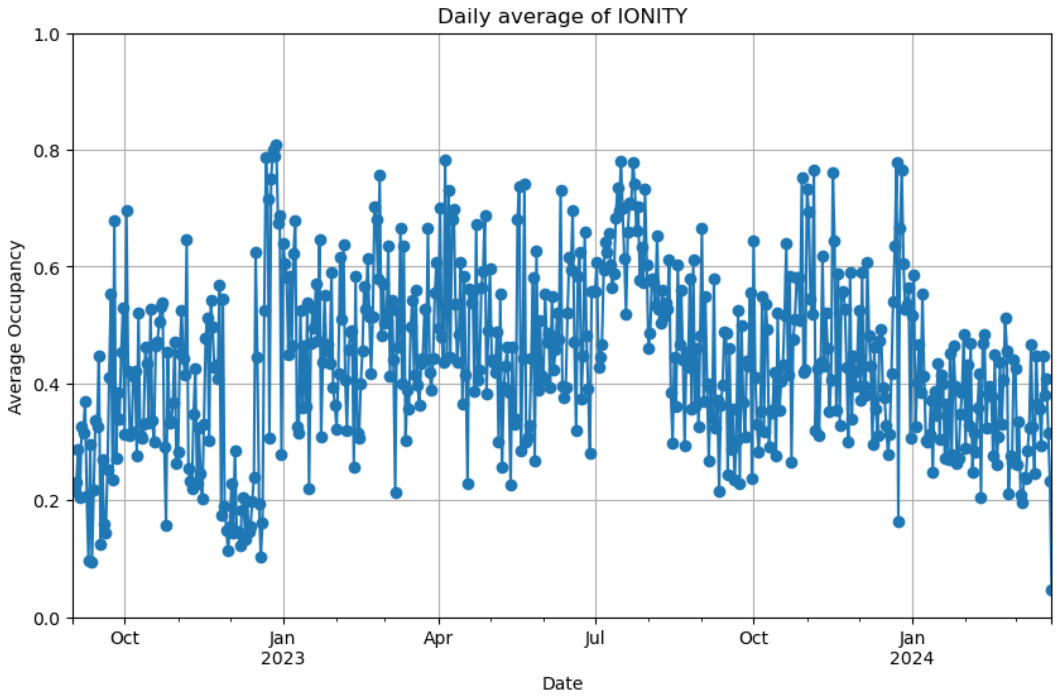

Figure 6: Mean values calculated on a daily basis across the entire dataset for the station IONITY.

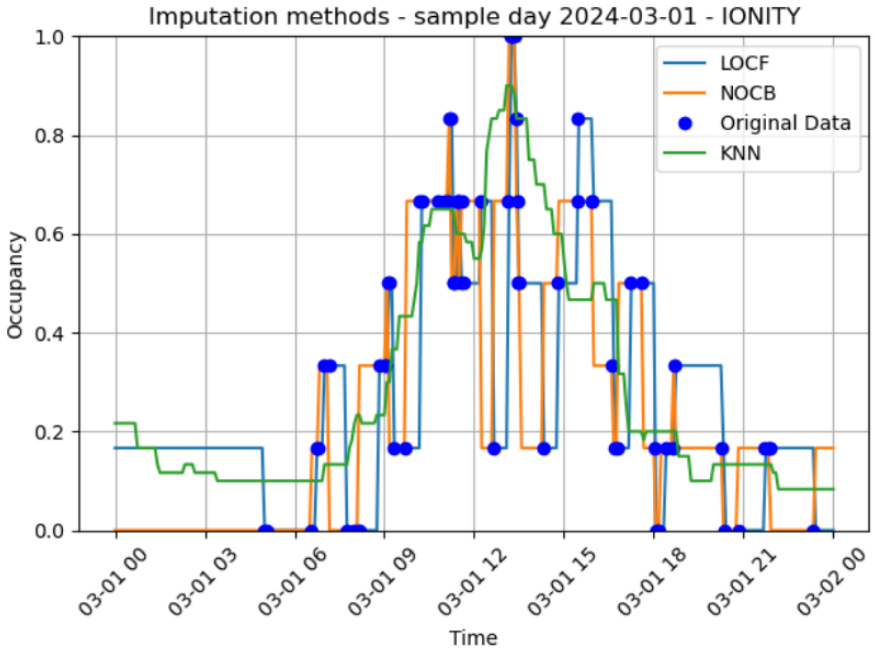

Figure 7: An overview of the diverse imputation methods on a sample day, 2024-03-01 on the IONITY dataset

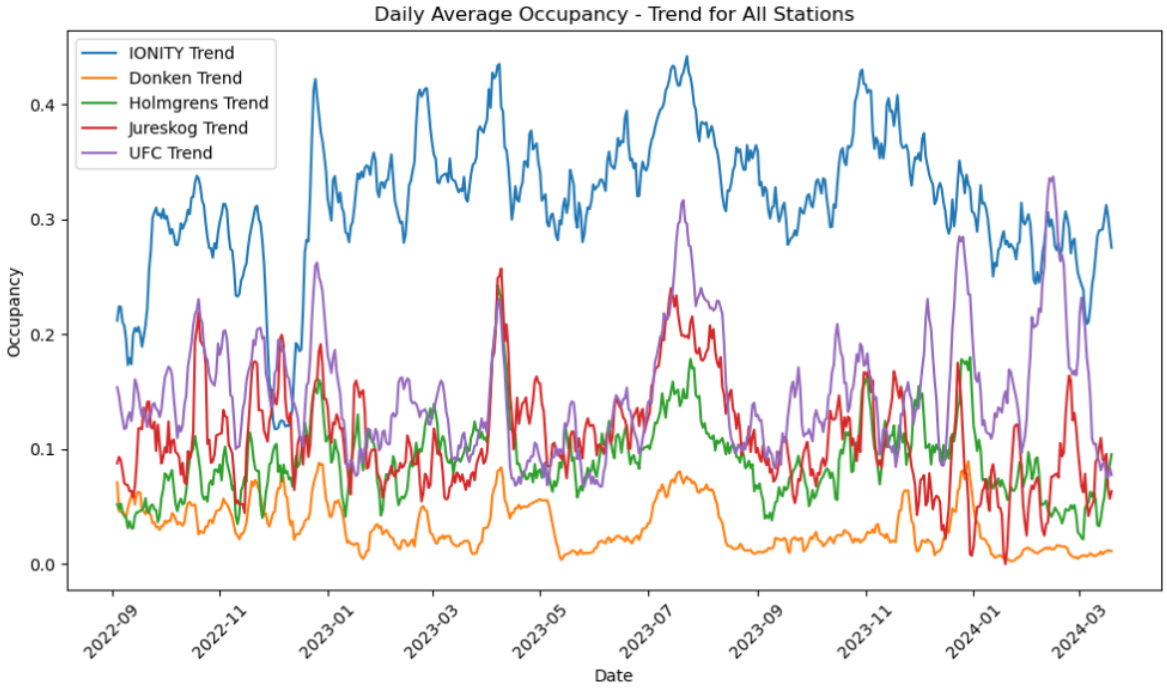

Figure 8: Visualization of the trend for the five charging stations.

positioned charging stations, all within a maximum detour threshold of 8 kilometers, and offering a power spectrum ranging from 50 kW to 350+ kW. Covering a distance of 235.1 kilometers, the anticipated journey duration along this route is estimated at 2 hours and 22 minutes.

In addition to the primary route, our attention was directed towards two urban cities, Varberg and Halmstad, each equipped with 14 and 7 chargers, respectively, as depicted in Figures 10 and 11. These areas have been selected due to their close proximity to one another.

Furthermore, another useful scenario for our analysis involves a cluster of 9 charging stations situated in the proximity of an exit in Värnamo, as illustrated in Figure 12, along the highway route defined before (Figure 9).

4.4 Horizontal Line

The "horizontal line" method is a simple yet insightful baseline approach in time series forecasting. This method involves extending a horizontal line from the last known value of the series into the future, effectively predicting that all future values will be equal to this last observed value. The elegance of the horizontal line method lies in its simplicity and the minimal assumptions it makes about the data. It serves as a useful benchmark because it provides a point of reference to evaluate the performance of more complex models.

4.5 ML Baseline Methods

An autocorrelation plot is generated to facilitate a visual assessment. The insights derived from this plot acts as a guideline for the selection of various starting parameters for the ML models used in this project, ensuring a tailored and effective application of these ML techniques on this data. Moreover, the hyperparameters specific to each model is optimized further with the use of grid search and the MSE criterion, as explained in Section 2.4.3.

In the context of 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 4.6, will be adopted. Initially, the MA, AR, and ARIMA models will be fitted to the first n datapoints (t to $t-n$) to predict the subsequent L datapoints (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

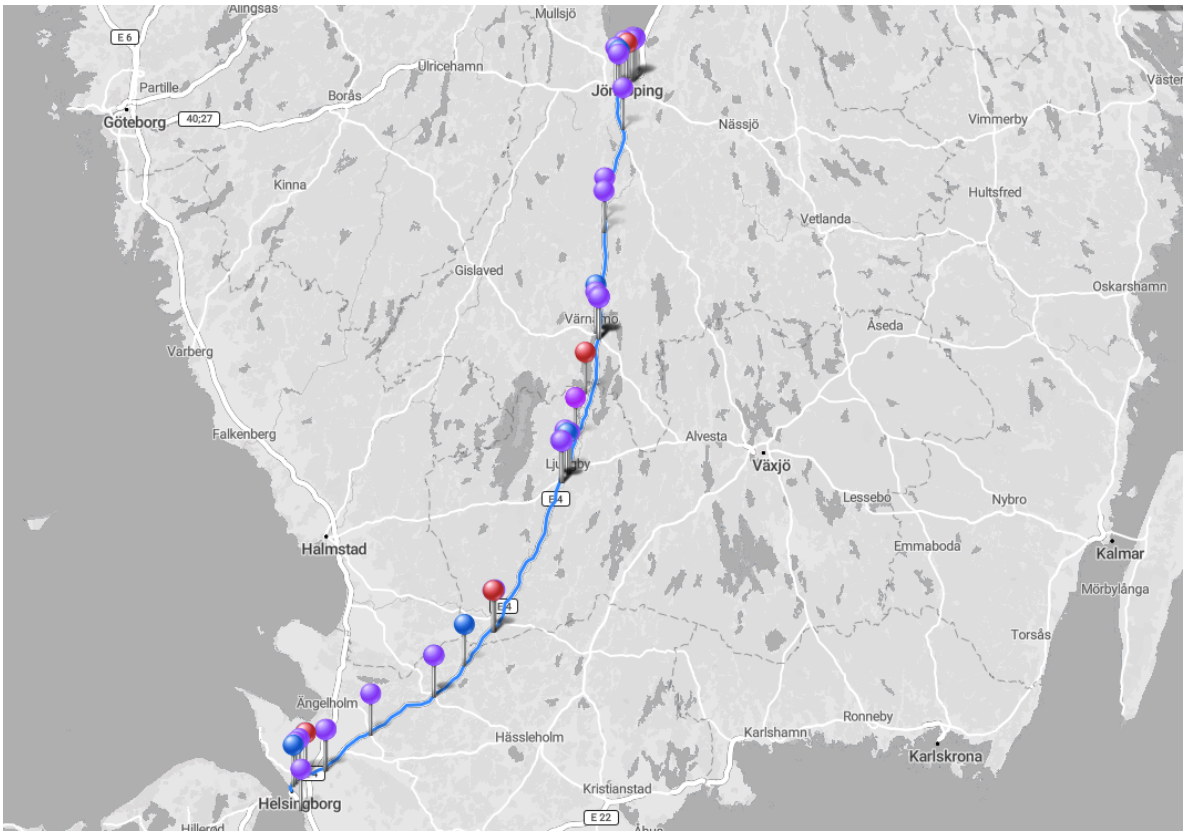

Figure 9: A captured image from ChargeFinder website illustrating the selected highway route pertinent to this thesis.

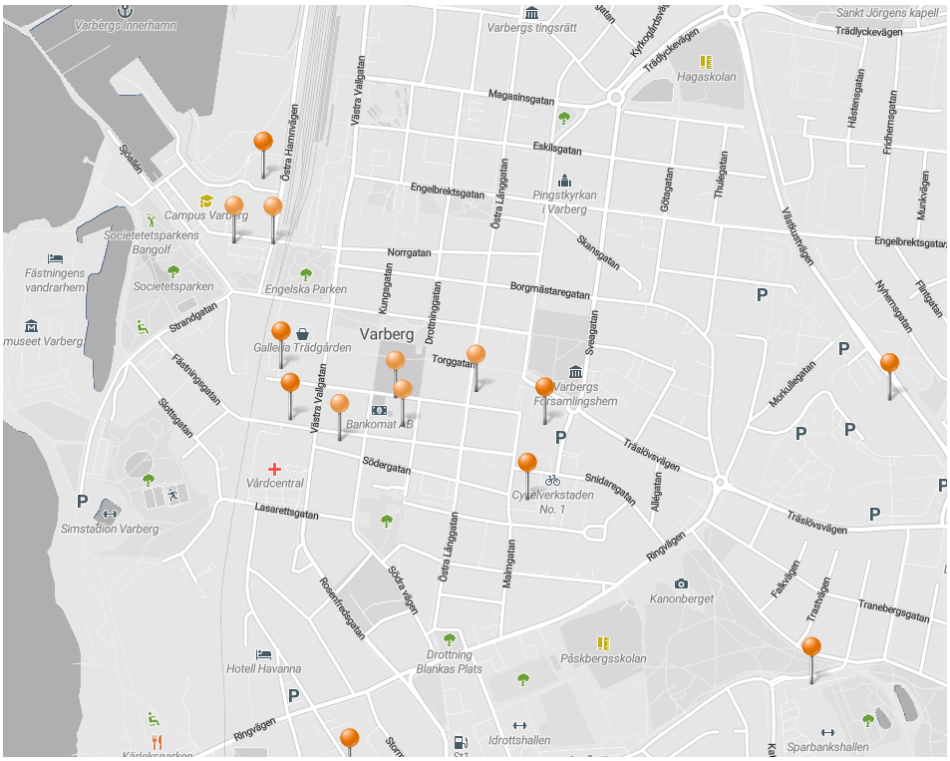

Figure 10: A captured image from ChargeFinder website illustrating one of the two city scenario to this thesis, namely Varberg.

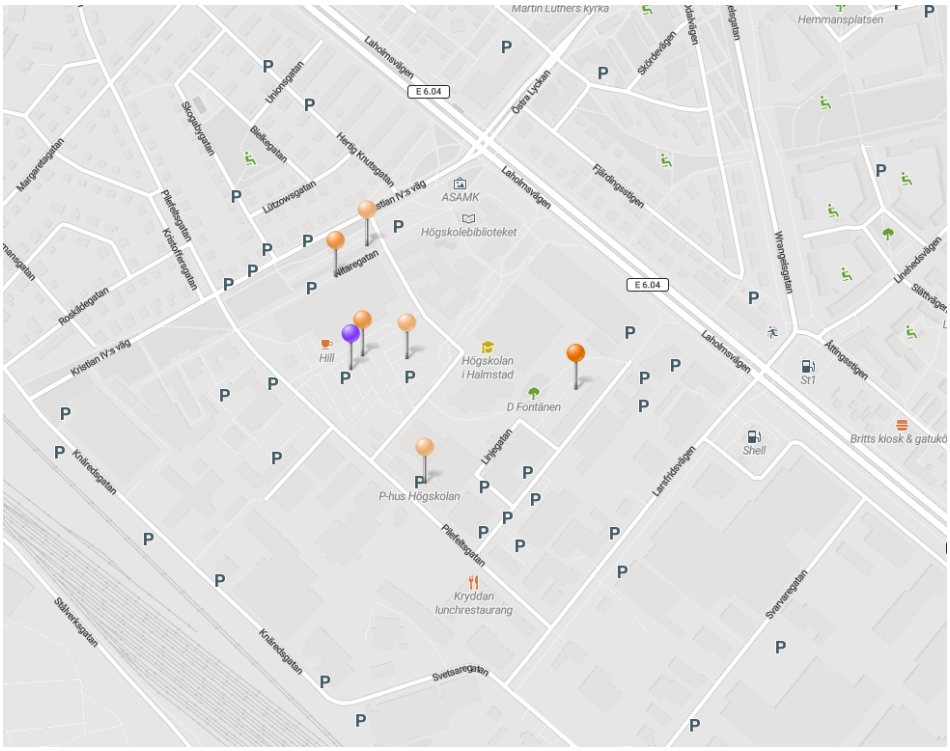

Figure 11: A captured image from ChargeFinder website illustrating one of the two city scenario to this thesis, namely Halmstad.

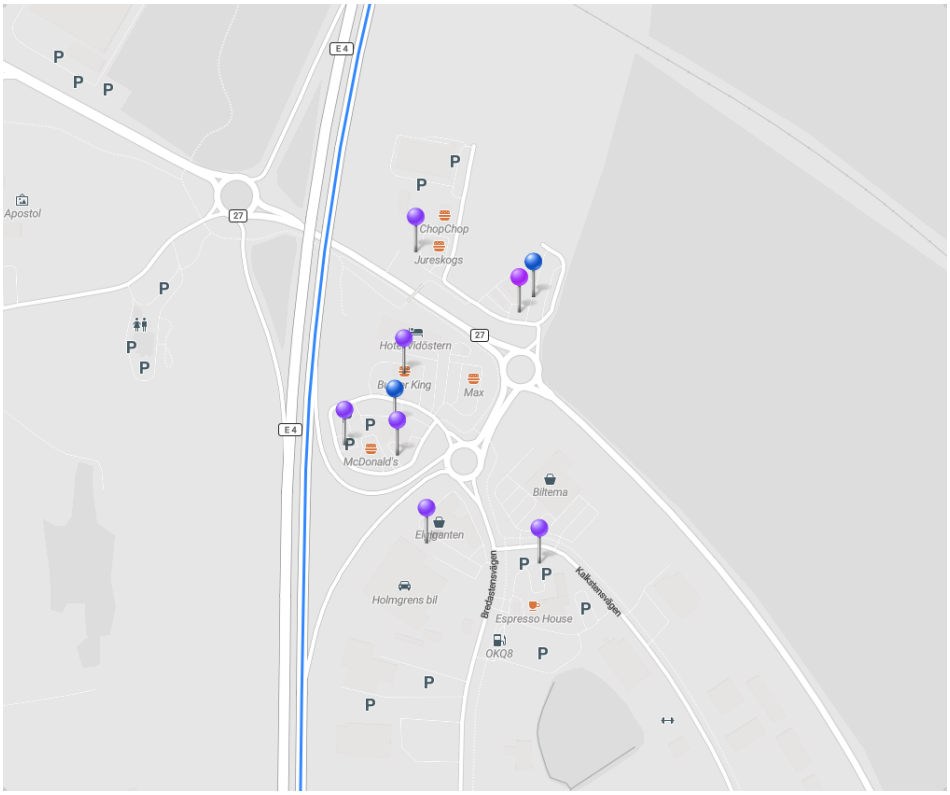

Figure 12: A captured image from ChargeFinder website illustrating an Exit in Värnamo, along the highway scenario depicted in Figure 9.

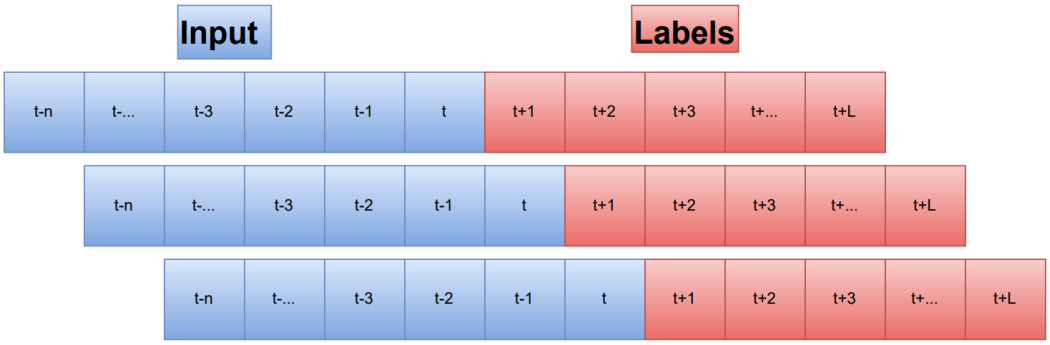

Figure 13: An illustration of three sequences resulting from the application of the sliding window technique twice. t = present time, n = Input sequence size, L = Output sequence size

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.

4.6 DL Baseline 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 4.5.

The DL models will be trained on data loaders, consisting of sequential inputs as depicted in Figure 13, where 3 sequence 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.

4.6.1 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 14.

4.6.2 Transformers

This project leverages the Transformer architecture as detailed in the seminal work, "Attention is All You Need"[24]. 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. Please note that all Transformer models in this project are trained with the dynamic learning rate explained in Equation 14.

As stated in Section 2.2.5, 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 streamlines the model's architecture by removing a redundant processing step, thereby allowing direct engagement with the numerical input data. The methodology for positional encoding incorporates sinusoidal or 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

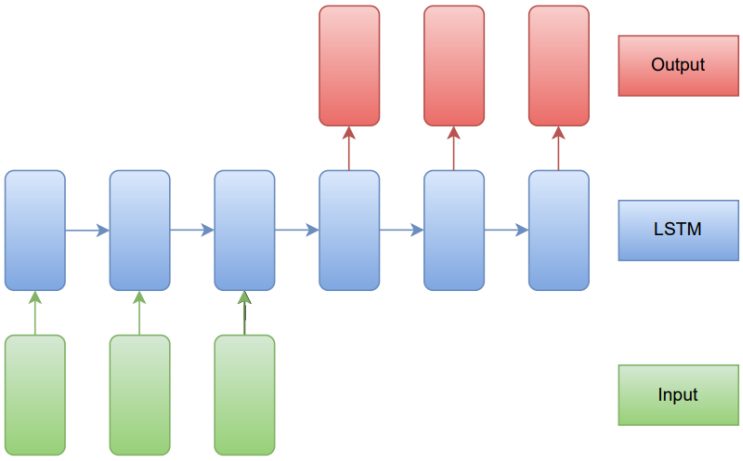

Figure 14: The projects LSTMs architecture configuration.

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 15: Visualization of the causal mask implementation.

the model's input. 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 18, and visually in Figure 15, 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 both the encoder and the decoder.

$$\text{Mask}(i, j) = \begin{cases} 0 & \text{if } i \leq j \\ -\infty & \text{otherwise} \end{cases} \tag{18}$$

As for the Encoder-only model, the inputs are treated as mentioned above, with a linear layer instead of an embedding layer, positional encoding, and causal masking the input. The difference between the Encoder-only model and the Encoder-Decoder is that instead of using the output of the encoder, namely, the learned representations of the input, as input to the decoder, it is sent through a fully connected layer which remaps the learned features to a sequence length which then becomes the predictions. Refer to Figure 16 for an overview of the model.

On the other hand, the Encoder-Decoder model uses the same input modifications as the Encoder-

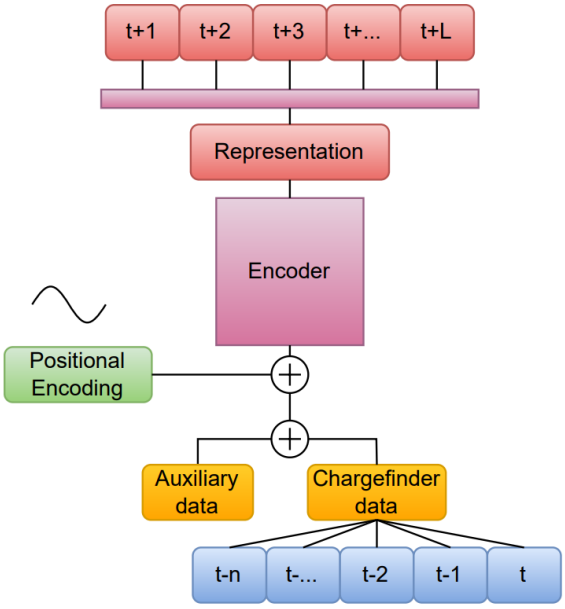

Figure 16: An overview schematic of the Encoder-only model.

only architecture. However, since the model now contains a decoder as well, 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 done by removing the softmax layer, usually indicative of classification tasks, and retaining only the linear layer as output. This modification aligns with the project's aim to produce continuous variable predictions rather than discrete categorizations. Please see Equation 19, 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.

$$\text{Output} = W_o H + b_o \tag{19}$$

Figure 17 shows an overview of the Encoder-Decoder model with the aforementioned changes implemented.

4.7 ST-GCN

The ST-GCN in this project is a hybrid network that consists of a GCN and a Transformer. The Transformer leverages an Encoder-Decoder architecture, as explained in 4.6.2. However, instead of inputting charging station occupancy over time for a charging station, the output data from a GCN will act as input to the Transformer. The final ST-GCN will resemble the network depicted in 18. 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, which is due to the design of the torch_geometric API[51]. The GCN output is then combined with the auxiliary data (common Features among the stations, i.e., weather information, holidays, etc.) and input into the Transformer. The Transformer will then predict a sequence for each node.

Architecture Overview:

- 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 regularization to extract hierarchical features from the input node features.

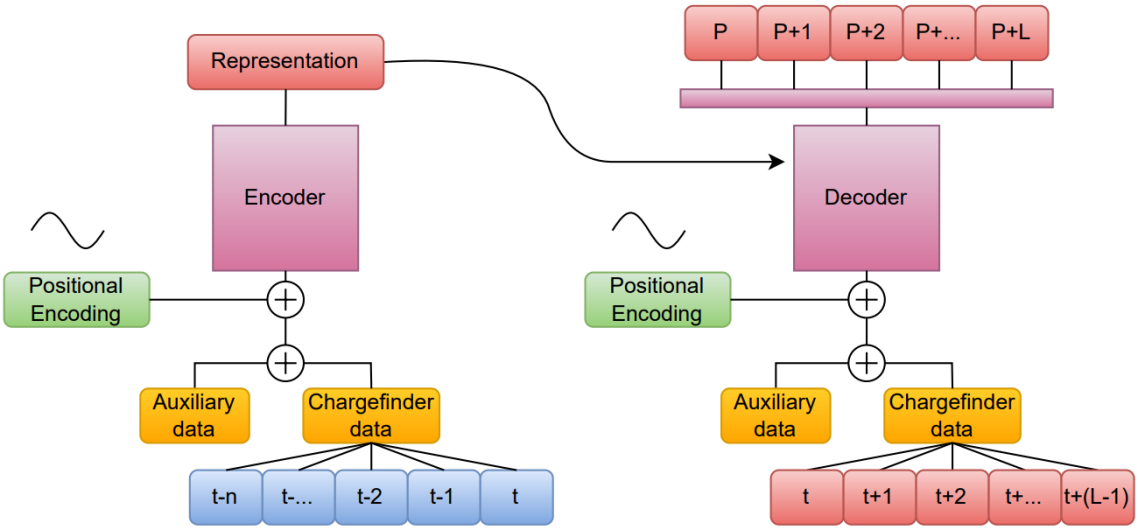

Figure 17: Visualization of the Encoder-Decoder structure of our Transformer.

- Transformer:
  - For temporal processing, ST-GCN utilizes a Transformer architecture, which employs sequence to sequence generation to forecast occupancy based on temporal the temporal aspect of the data.

5 Results

In determining the appropriate input window size for our analysis, the autocorrelation plot provided valuable insights, see Figure 19, indicating a distinct daily seasonal pattern. This pattern highlighted a strong dependency up to the preceeding 3 hours (36 datapoints), supplemented by a seasonal correlation with equivalent times from preceding days, all showcasing autocorrelation values around 0.5. Consequently, it was decided to adopt an input window spanning the last 2 days, translating to 576 data points ($n = 576$). Additionally, in alignment with the forecasting objectives, the output window was set to 36 data points, corresponding to the next 3 hours. Moreover, selecting a 3-hour forecasting horizon closely corresponds to the travel duration of the Helsingborg-Jönköping route, illustrated in Figure 9. This route typically takes 2 hours and 22 minutes, defining our prediction window to cover the entirety of this journey.

5.1 Routes

Currently, the dataset comprises data solely from the nine stations located in Värnamo, see Figuer 12, named UFC, IONITY, Jureskogs_vattenfall, Donken, Holmgrens, BurgerKing, OKQ8, ICA, and Jureskog_Recharge. However, due to limited data availability for certain stations—such as Jureskogs_Recharge, which only spans two months—some charging stations were excluded from the analysis. Therefore, the Värnamo network analyzed in this project consists of five charging stations: UFC, IONITY, Jureskogs_Vattenfall, Donken, and Holmgrens. All of them have data spanning from September 2022 to March 2024.

Up to this point, preliminary experiments from the models have been conducted on solely the IONITY station data. It's important to clarify that the results from the models presented below are all derived from the IONITY dataset, given its comprehensive data coverage, allowing for robust analysis and comparison. Further experiments will be executed on all datasets in the near future.

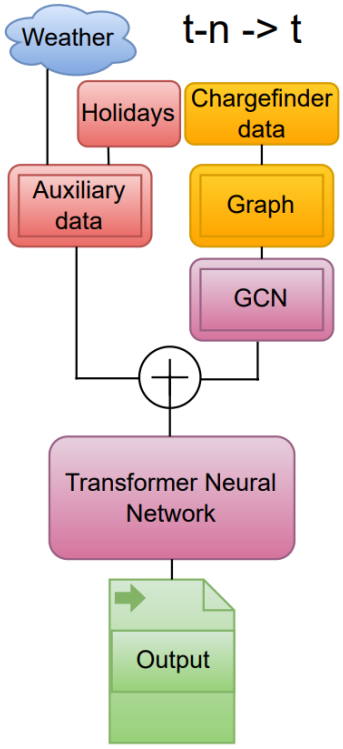

Figure 18: A network diagram for the ST-GCN in this thesis.

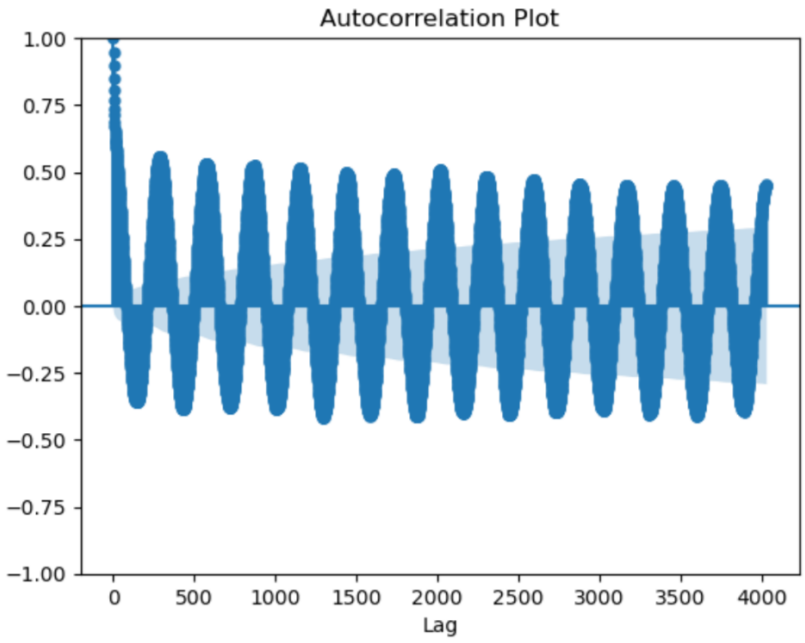

Figure 19: Autocorrelation plot of the data from our test station.

Model	$\overline{\text{MSE}}$	Model-Type
Horizontal line	0.0215	Linear
MA	0.0392	ML
AR	0.0223	ML
ARIMA	0.0320	ML
LSTM	0.0134	DL
Transformer-Encoder-Only	0.0182	DL
Transformer-Encoder-Decoder	**0.0122**	DL
ST-GCN (Transformer)	-	DL

Table 2: Collection of the average MSE values on IONITY dataset from all models utilized in this project.

5.2 MA

As for the MA, the parameter q was set to 36, aligning with the requirement to predict 36 timesteps into the future. This allows us to prevent the model from producing a constant forecast for the timesteps after $t+q$, and up to $t+L$. To better explain, a model with a q value of 4, when predicting 10 datapoints into the future, it would forecast the average for the first 4 datapoints and maintain a constant prediction thereafter, i.e., for the datapoints spanning from $t+4$ to $t+10$. Moreover, the choice of setting the q parameter to 36 also stems from the notable correlations observed between the current time t and the preceding $t-36$ timesteps, as depicted in the correlation plot shown in Figure 19. Furthermore, as discussed in Section 4.5, this parameter, alongside the parameters for AR and ARIMA, will undergo refinement through grid search methodology.

Overall, the MA model scored an average MSE value of 0.0392 throughout the dataset, as seen in Table 2. This result falls below the average MSE score of all the other models.

5.3 AR

Regarding the AR, it's p parameter is set to 36, for the same reason as the q parameter for the MA model. The AR model yielded an average MSE value of 0.0223 across the dataset, as illustrated in Table 2. Although this performance is better than the MA, it falls below the average MSE score attained by the other models.

5.4 ARIMA

The ARIMA model, incorporating AR, differencing, and MA components, have the parameters p and q for the AR and MA algorithms both initialized at 36. The I parameter, instead, is set to 0, signifying that the ARIMA model utilizes the raw, undifferenced dataset, accepting the trands and seasonality.

As for the results of the ARIMA model, it demonstrates an average MSE value of 0.0320 across the dataset., as detailed in Table 2. While this performance surpasses that of the MA model, it still got a worse result than the AR model and DL models.

5.5 LSTM

Introducing the LSTM model as the first baseline DL method yielded impressive results, outperforming all the ML models as well as the horizontal line benchmark. With an average MSE score of 0.0134, the LSTM emerged as the top-performing model in this comparative analysis so far. Please see Table 3 for a complete table containing the parameters for the DL models.

Additionally, Figure 20 provides a visual representation of the predictive performance of the LSTM model. It showcases the predicted sequences alongside their corresponding true labels for 16 randomly selected sequences at test time. This visualization serves to illustrate the LSTM model's performance in capturing the patterns and trends in the occupancy data. Upon examining the figure, it's evident that the forecasts generally trend in the correct direction. Nonetheless, the predicted values tend to form relatively straight lines, akin to those produced by an Encoder-only model. This characteristic

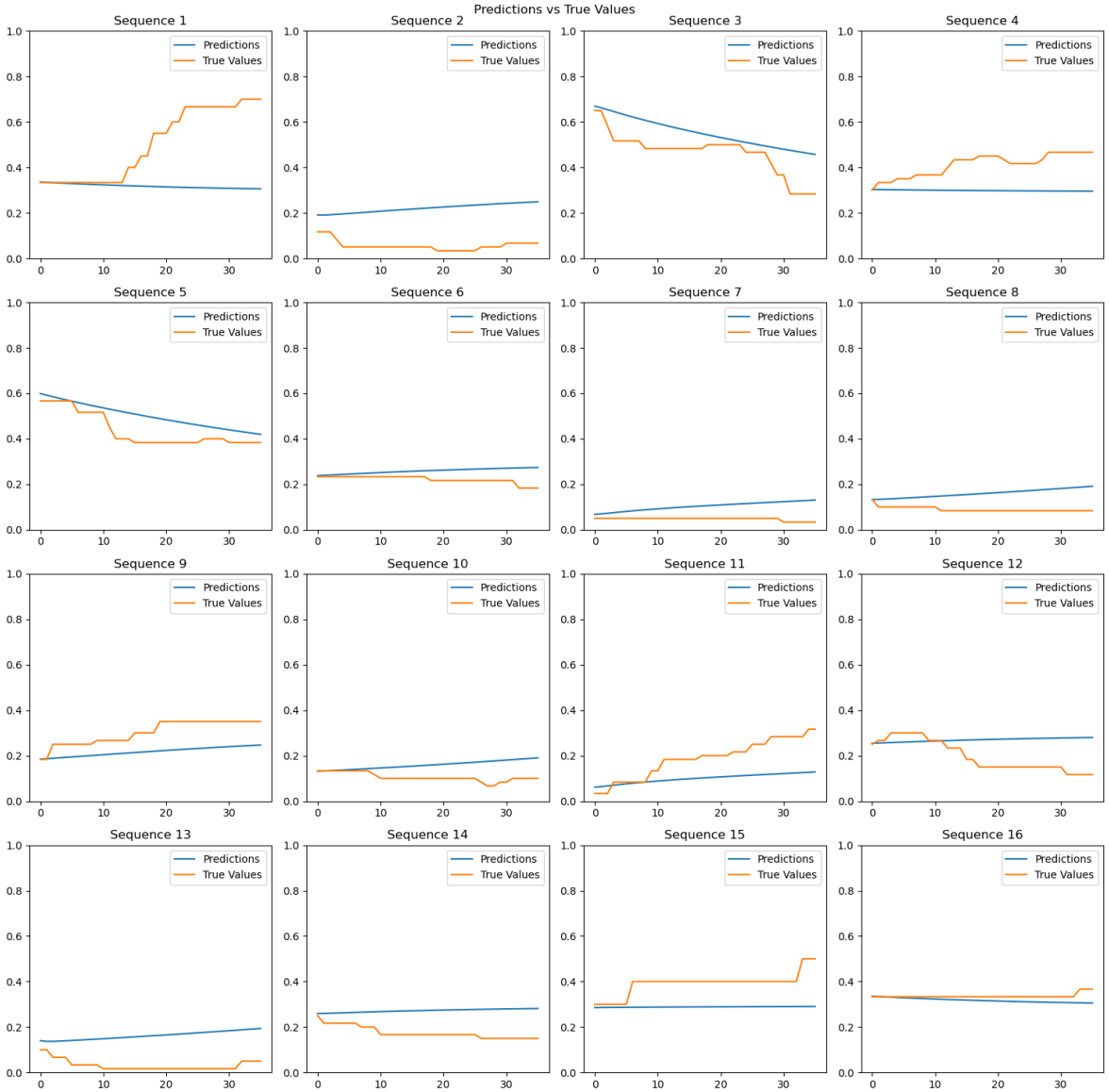

Figure 20: An overview of the LSTM model's predictions over 16 random sequences.

likely contributes to the model's Mean Squared Error (MSE) being marginally higher than that of the Transformer Encoder-Decoder model, which exhibits a degree of nonlinearity in its predictions.

5.6 Transformer

The initial stage focuses on deploying the simplest variant of the model, an Encoder-only Transformer. Following this, the investigation progresses to a second phase, which involves integrating a decoder to explore the benefits of a full Encoder-Decoder framework.

5.6.1 Encoder-only model

The performance of the Encoder-only Transformer model outperformed the ML methods and the simplistic linear Horizontal line benchmark. However, the final MSE score achieved by the Encoder-only architecture on the IONITY dataset was 0.0182, which lags behind the performance of both the MultiStepLSTM and the Transformer model equipped with both encoder and decoder components, amplifying the significance of using a Transformer Encoder-Decoder architecture for this project.

Parameter	LSTM	Transformer-Enc	Transformer-EncDec	ST-GCN
Starting Learning Rate	0.005	0.001	0.001	-
Scheduler	ReduceLROnPlateau	Equation 14	Equation 14	-
Early Stopping, patience	Yes, patience = 50	Yes, patience = 50	Yes, patience = 50	-
Optimizer	Adam	Adam	Adam	-
criterion	MSE	MSE	MSE	-
Dropout Rate	-	0.1	0.1	-
Batch size	8	64	64	-
Activation Function		gelu	gelu	-
Hidden Layers	3	1	1	-
Hidden Layer Size	1	10 (40)	10 (40)	-
Number Of Heads	-	1	1	-
Input Features	Univariate	Univariate	Univariate	-
Output Features	Univariate	Univariate	Univariate	-
Input sequence length	576	576	576	-
Output sequence length	36	36	36	-
Trainable Parameters	-	-	-	-

Table 3: Collection of the parameters for the DL models used in this project as of now.

Furthermore, Figure 21 provides a visual representation of the predictive performance of the Transformer Encoder-only model. The outcomes and forecasts generated by this model closely resemble those obtained from the LSTM model. However, the directionality of the Encoder-only model's predictions aligns slightly less accurately with the observed data trend compared to the LSTM predictions.

5.6.2 Encoder-Decoder

Remarkably, this Encoder-Decoder model achieved the highest MSE score among all the models, with a MSE= 0.0122. Notably, during the training of the Transformer, a technique which involves providing the decoder with the true labels of its previous predictions is used. This practice aims to facilitate convergence by aiding the model in making accurate predictions. However, intriguingly, when trained with this technique, the model yielded an error of 0.0128. In contrast, when trained by solely refeeding the model's last made predictions — it required significantly more time for training, as expected. Nevertheless, this alternative training approach resulted in a marginally improved MSE score of 0.0122.

Moreover, Figure 22 offers a visual depiction of the Transformer Encoder-Decoder model's predictive capabilities. Upon examination, it becomes evident that this model surpasses both the Encoder-only and LSTM models in adapting to occupancy changes. It consistently generates non-linear predictions that closely align with future trends in nearly all scenarios, unlike the other deep learning models, which tend to favor linear predictions.

5.7 Our ST-GCN

Due to delays encountered while programming the Transformer architecture, the ST-GCN has, for now, been implemented with an LSTM instead. Currently, the ST-GCN only takes one input feature per node. The GCN then outputs a tensor where each node has eight features, which is then input into the LSTM. The LSTM also has a hidden size of eight and outputs nodes with one feature representing occupancy. However, the ST-GCN is not yet optimized, and the project has no results to show so far.

5.8 Agent

No results so far

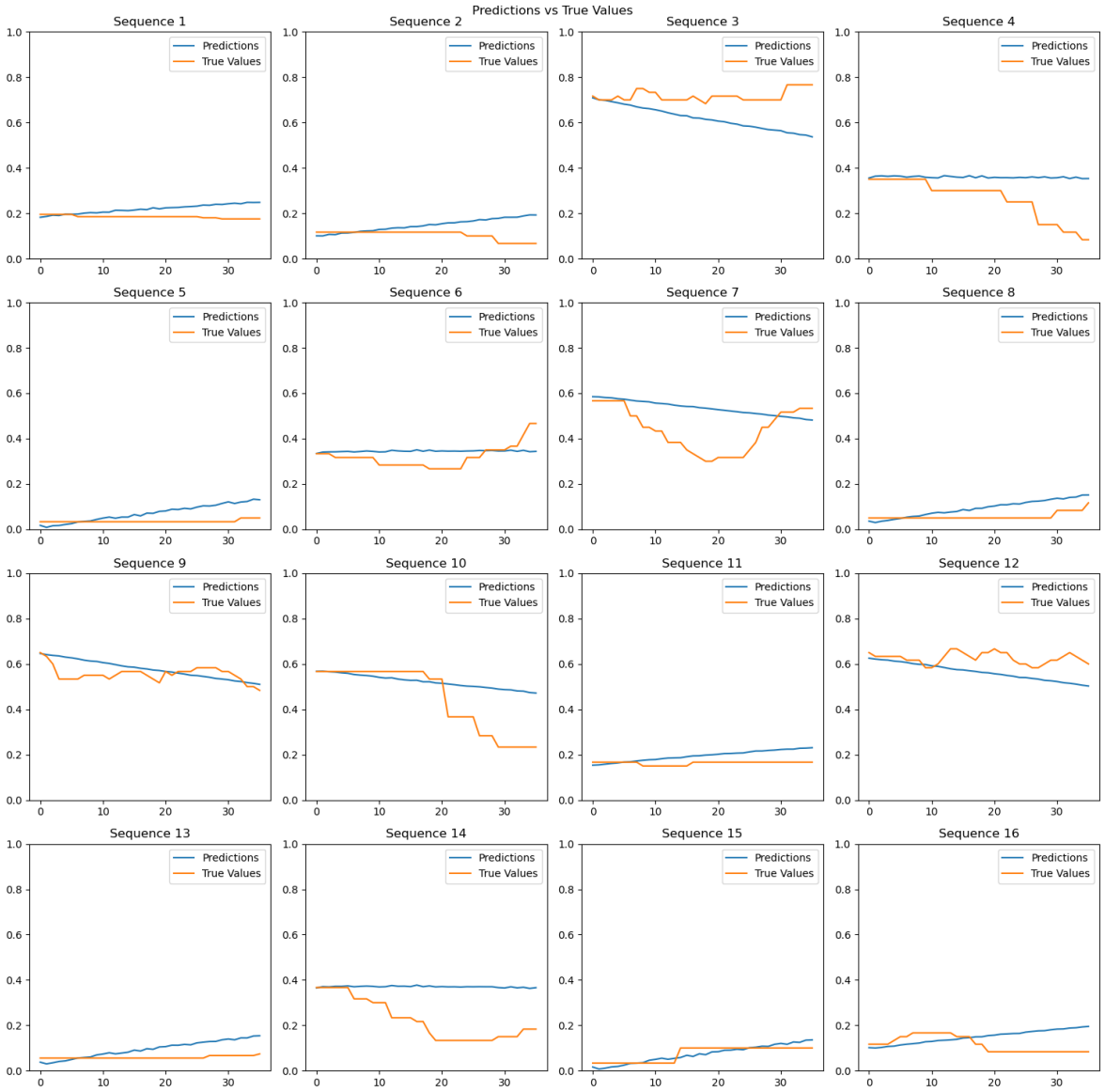

Figure 21: An overview of the Transformer Encoder-only model's predictions over 16 random sequences.

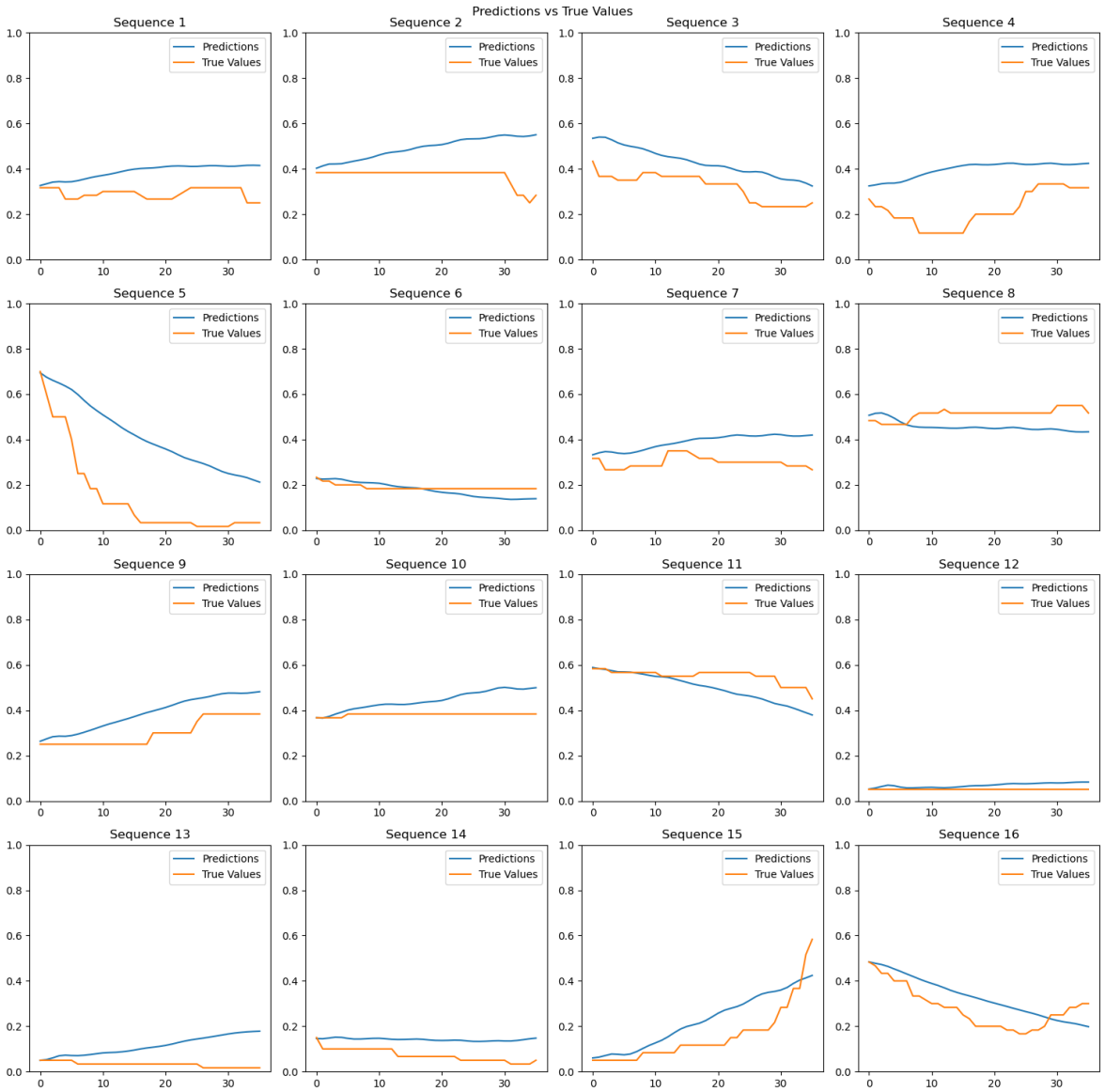

Figure 22: An overview of the Transformer Encoder-Decoder model's predictions over 16 random sequences.

6 Conclusion

This thesis has developed a method to forecast EV charging station occupancy and to offer route recommendations using DL. The baseline methods are complete and produce results that are somewhat within expectations. The ST-GCN is in its finishing stages. Currently, the ST-GCN is using an LSTM as its temporal component. However, the plan is to exchange the LSTM with a Transformer after the halftime seminar. There has been no development in creating the DRL agent. The project is overall proceeding as planned and is expected to be completed on time.

References

[1] ChargeFinder. Data Provided for Neural Network Training; 2024. Data provided by contact at ChargeFinder. Personal communication.

[2] IEA. Electric Vehicles; 2023. Accessed on 2023-11-24. Available from: https://www.iea.org/energy-system/transport/electric-vehicles.

[3] Volvo Cars; 2021. Available from: https://www.media.volvocars.com/global/en-gb/media/pressreleases/277409/volvo-cars-to-be-fully-electric-by-2030.

[4] Automotive News Europe; 2022. Available from: https://europe.autonews.com/automakers/vw-brand-will-be-electric-only-europe-2033.

[5] KIA Motors;. Available from: https://press.kia.com/ie/en/home/media-resouces/press-releases/2021/Sustainable-Mobility-Solutions-Provider1.html.

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

[7] 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. Accessed on 2023-11-24. Available from: https://ec.europa.eu/commission/presscorner/detail/en/IP_22_6462.

[8] Sao A, Tempelmeier N, Demidova E. Deep Information Fusion for Electric Vehicle Charging Station Occupancy Forecasting. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC); 2021. p. 3328-33.

[9] Zamee MA, Han D, Cha H, Won D. Self-supervised online learning algorithm for electric vehicle charging station demand and event prediction. J Energy Storage. 2023 Nov;71(108189):108189.

[10] Lee B, Lee H, Ahn H. Improving load forecasting of electric vehicle charging stations through missing data imputation. Energies. 2020 Sep;13(18):4893.

[11] Ma TY, Faye S. Multistep electric vehicle charging station occupancy prediction using hybrid LSTM neural networks. Energy. 2022;244:123217. Available from: https://www.sciencedirect.com/science/article/pii/S0360544222001207.

[12] Luo R, Song Y, Huang L, Zhang Y, Su R. AST-GIN: Attribute-Augmented Spatiotemporal Graph Informer network for electric Vehicle charging station availability forecasting. Sensors (Basel). 2023 Feb;23(4).

[13] Su S, Li Y, Chen Q, Xia M, Yamashita K, Jurasz J. Operating status prediction model at EV charging stations with fusing spatiotemporal graph convolutional network. IEEE Trans Transp Electrif. 2022:1-1.

[14] Hassani H, Kalantari M, Ghodsi Z. 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. 2019 Dec;2(4):457-67.

[15] Lee B, Lee H, Ahn H. Improving Load Forecasting of Electric Vehicle Charging Stations Through Missing Data Imputation. Energies. 2020;13(18). Available from: https://www.mdpi.com/1996-1073/13/18/4893.

[16] Alsharef A, Aggarwal K, Sonia, Kumar M, Mishra A. Review of ML and AutoML solutions to forecast time-series data. Archives of Computational Methods in Engineering. 2022;29(7):5297-311.

[17] Ivanovski Z, Milenkovski A, Narasanov Z. Time series forecasting using a moving average model for extrapolation of number of tourist. UTMS Journal of Economics. 2018;9(2):121-32.

[18] Siami-Namini S, Tavakoli N, Namin AS. A comparison of ARIMA and LSTM in forecasting time series. In: 2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE; 2018. p. 1394-401.

[19] Elsaraiti M, Merabet A. Solar Power Forecasting Using Deep Learning Techniques. IEEE Access. 2022;10:31692-8.

[20] Leeraksakiat P, Pora W. 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; 2020. .

[21] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9(8):1735-80.

[22] Li AW, Bastos GS. Stock Market Forecasting Using Deep Learning and Technical Analysis: A Systematic Review. IEEE Access. 2020;8:185232-42.

[23] Sagheer A, Kotb M. Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing. 2019;323:203-13. Available from: https://www.sciencedirect.com/science/article/pii/S0925231218311639.

[24] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems. 2017.

[25] Tong J, Xie L, Yang W, Zhang K, Zhao J. Enhancing time series forecasting: A hierarchical transformer with probabilistic decomposition representation. Information Sciences. 2023;647:119410. Available from: https://www.sciencedirect.com/science/article/pii/S0020025523009957.

[26] Wu N, Green B, Ben X, O'Banion S. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case; 2020.

[27] Lim B, Arık S Loeff N, Pfister T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting. 2021;37(4):1748-64. Available from: https://www.sciencedirect.com/science/article/pii/S0169207021000637.

[28] Shaw P, Uszkoreit J, Vaswani A. Self-Attention with Relative Position Representations; 2018.

[29] Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35; 2021. p. 11106-15.

[30] Woo G, Liu C, Sahoo D, Kumar A, Hoi S. Etsformer: Exponential smoothing transformers for time-series forecasting. arXiv preprint arXiv:220201381. 2022.

[31] Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang YX, et al. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in neural information processing systems. 2019;32.

[32] Lara-Benítez P, Gallego-Ledesma L, Carranza-García M, Luna-Romera JM. Evaluation of the transformer architecture for univariate time series forecasting. In: Advances in Artificial Intelligence: 19th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2020/2021, Málaga, Spain, September 22–24, 2021, Proceedings 19. Springer; 2021. p. 106-15.

[33] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:160902907. 2016.

[34] Almasan P, Suárez-Varela J, Rusek K, Barlet-Ros P, Cabellos-Aparicio A. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. Comput Commun. 2022 Dec;196:184-94.

[35] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:13125602. 2013.

[36] Li Y. Deep reinforcement learning: An overview. arXiv preprint arXiv:170107274. 2017.

[37] Lee KB, A Ahmed M, Kang DK, Kim YC. Deep Reinforcement Learning Based Optimal Route and Charging Station Selection. Energies. 2020;13(23). Available from: https://www.mdpi.com/1996-1073/13/23/6255.

[38] Zhang C, Liu Y, Wu F, Tang B, Fan W. Effective Charging Planning Based on Deep Reinforcement Learning for Electric Vehicles. IEEE Transactions on Intelligent Transportation Systems. 2021;22(1):542-54.

[39] Voyant C, Notton G, Kalogirou S, Nivet ML, Paoli C, Motte F, et al. Machine learning methods for solar radiation forecasting: A review. Renewable Energy. 2017;105:569-82. Available from: https://www.sciencedirect.com/science/article/pii/S0960148116311648.

[40] del Real AJ, Dorado F, Durán J. Energy Demand Forecasting Using Deep Learning: Applications for the French Grid. Energies. 2020;13(9). Available from: https://www.mdpi.com/1996-1073/13/9/2242.

[41] Zhang C, Liu Y, Wu F, Tang B, Fan W. Effective Charging Planning Based on Deep Reinforcement Learning for Electric Vehicles. IEEE Transactions on Intelligent Transportation Systems. 2021;22(1):542-54.

[42] Ogunsanya M, Isichei J, Desai S. Grid search hyperparameter tuning in additive manufacturing processes. Manufacturing Letters. 2023;35:1031-42. 51st SME North American Manufacturing Research Conference (NAMRC 51). Available from: https://www.sciencedirect.com/science/article/pii/S221384632300113X.

[43] Radzi SFM, Karim MKA, Saripan MI, Rahman MAA, Isa INC, Ibahim MJ. Hyperparameter Tuning and Pipeline Optimization via Grid Search Method and Tree-Based AutoML in Breast Cancer Prediction. Journal of Personalized Medicine. 2021;11(10). Available from: https://www.mdpi.com/2075-4426/11/10/978.

[44] Sanchez OR, Repetto M, Carrega A, Bolla R. Evaluating ML-based DDoS detection with grid search hyperparameter optimization. In: 2021 IEEE 7th International Conference on Network Softwarization (NetSoft). IEEE; 2021. p. 402-8.

[45] Mantovani RG, Rossi ALD, Vanschoren J, Bischl B, de Carvalho ACPLF. Effectiveness of Random Search in SVM hyper-parameter tuning. In: 2015 International Joint Conference on Neural Networks (IJCNN); 2015. p. 1-8.

[46] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of machine learning research. 2012;13(2).

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

[48] Riksdagen. Riksdagens holidays; 1989. Lag (1989:253) om allmänna helgdagar. https://www.riksdagen.se/sv/dokument-och-lagar/dokument/svensk-forfattningssamling/lag-1989253-om-allmanna-helgdagar$_sfs-1989-253$/.

[49] Google LLC. Google Maps; 2024. Accessed: 2024-04-06. https://www.google.com/maps.

[50] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30.

[51] Fey M, Lenssen JE. Fast Graph Representation Learning with PyTorch Geometric. GitHub; 2019. https://github.com/rusty1s/pytorch_geometric.