Aggregation and Prediction of Energy Consumption Data
What is the Aggregation Level at which a Graph Neural Network Performs Optimally?

Lennard Timp

Supervisor(s): dr. ir. L. Cavalcante Siebert, ir. S. Kuilman

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Lennard Timp
Final project course: CSE3000 Research Project
Thesis committee: prof. dr. M. M. de Weerdt, dr. ir. L. Cavalcante Siebert, ir. S. Kuilman,

Abstract

Electrical load forecasting, namely short-term load forecasting, is essential to power grids' safe and efficient operations. The need for accurate short-term load forecasting becomes increasingly pressing with increased renewable energy sources, which are stochastic in their power supply. Most forecasting models are focused on the temporal information for predictions, ignoring the spatial information of neighbouring houses. However, as neighbouring houses are often under similar circumstances, such as weather and holiday conditions, predictions could benefit from this information. Moreover, aggregating electric loads could further improve the accuracy of predictions, as the loads become less stochastic when aggregated. This paper looks at Graph WaveNet, which can capture the hidden spatial dependencies of different houses without needing any prior knowledge about the houses, such as geographic information. The framework is compared against a baseline on different aggregation levels. The results show that the framework can benefit from aggregating the residential electrical loads and improves over the baseline on all aggregation levels.

1 Introduction

One of the challenges modern power grids face is an increase in renewable energy sources, which are stochastic in their power supply. This calls for accurate demand predictions to provide an optimised supply. Due to the hierarchical nature of power grids, high demand prediction performance at lower demand aggregations allows for optimal performance. When load forecasting is inaccurate, it can cause a vast financial burden to a utility operating in time-ahead power markets. An accuracy improvement of 1% is estimated to save a cost of £ 10 million per year for the United Kingdom (UK) power system [1].

When talking about the problem of load forecasting, there are three different categories, depending on the forecasting horizon: long-term (years ahead), mid-term (weeks to months ahead), and short-term (minutes to hours ahead). Short-term load forecasting (STLF) is often used to help real-time energy dispatching [2]. However, accurate STLF can be very challenging. Demand and supply uncertainties are significantly impacting modern power grids, with renewable energy sources providing a stochastic energy supply. These sources rely on unpredictable factors like weather conditions. At the same time, there is a sharp increase in the number of electrical appliances demanding energy, such as electric vehicles, making the power system increasingly complex [3].

Different types of methods have been proposed for the STLF problem. These methods can be categorised into statistical methods and machine learning methods. Statistical methods such as autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and exponential smoothing have been applied to short-term load forecasting (STLF) [4; 5; 6; 7; 8]. These statistical methods are low-cost and easily applicable techniques to capture the mathematical relationships between time series. However, these methods fail to remain accurate when a longer prediction horizon is needed [9]. Machine learning algorithms can capture complex non-linear relationships. Support vector regression (SVR), kernel-based methods and feed-forward neural networks (FNN) have shown successful application for STLF [10; 11; 12].

Furthermore, these machine learning methods are based on capturing the temporal relationships between time series, i.e. the historical data. They are thus unable to capture the spatial relationships between different houses. There are low-dimensional, sparse relationships between houses [13]. For example, houses in the same neighbourhood might have similar loads as they experience the same weather conditions and holidays. Houses can also have similar loads that are not based on their geographical similarity, but they might have the same electrical appliances and number of residents. For this reason, non-Euclidean pairwise correlations might better capture the similarities between houses, than Euclidean pairwise correlations, such as geographic locations.

Research into capturing the latent spatial relationships between houses is lacking. A data mining technique based on knowledge discovery in databases (KDD) has been used by Wu and Lu to capture the association between spatial data and load changes [14]. The procedure automatically determines the preferential 'scores' of land use changes. However, this technique uses additional information, such as land use, to capture the spatial relationships between houses.

This paper will analyse the performance of a graph neural network (GNN) based on the Graph WaveNet (GWN) model at different aggregation levels [15]. Specifically, this paper will focus on the following question: *'What is the aggregation level at which a Graph Neural Network performs optimally?'* Unlike other machine learning methods that were previously used for electric load forecasting, the GWN model does not need additional information besides the historical loads. As this is often difficult to obtain in real-life, GWN is easier to implement in real-life situations.

In section 4.4, we will compare GWN against an ARIMA model. To compare the performances at different aggregation levels, we will use the mean absolute error (MAE), the mean absolute percentage error (MAPE) and the root mean squared error (RMSE), as done in previous studies [16]. Then we will also look at a detailed prediction of the model at different aggregation levels. Finally, we will analyse the training times of Graph WaveNet, compared to an ARIMA model.

2 Research background

In section 2.1, we will describe the ARIMA model. Then section 2.2 will describe the GNN. Finally, in section 2.3 we

will talk about adjacency matrix determination.

2.1 ARIMA

In time series analysis, the autoregressive integrated moving average (ARIMA) model is a powerful tool for STLF. ARIMA is a versatile model that combines autoregressive (AR), differencing (I), and moving average (MA) components. The AR component captures the linear relationship between the current value of the variable and its lagged (i.e. prior) values. The MA component represents the linear dependence between the variable and the residual error from a moving average model applied to its lagged values. The I component indicates the order of differencing applied to the time series, which removes trends and seasonality. Combining these components allows the ARIMA model to capture complex dependencies within the temporal data.

Mathematically, the ARIMA model of order (p, d, q) can be defined by the following equations:

$$\varphi(L)(1-L)^d y_t = \theta(L)\varepsilon_t \tag{1}$$

$$(1-\sum_{i=0}^{p}\varphi_i L^i)(1-L)^d = (1+\sum_{j=1}^{q}\theta_j L^j)\varepsilon_t \tag{2}$$

$$\begin{aligned} Y_t = {} & \phi_0 + \phi_i y_{t-1} + \phi_i y_{t-2} + \cdots + \phi_p y_{t-p} \\ & + \varepsilon_t - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2} - \cdots - \theta_q\varepsilon_{t-q} \end{aligned} \tag{3}$$

Here, y_t and ε_t and the real value and random error at the t^{th} time step, respectively. $\varphi_i, i \in \{1,\ 2,\ \ldots\ ,\ p\}$ and $\theta_j, j \in \{1,\ 2,\ \ldots\ ,\ q\}$ are model parameters. p, d and q are positive integers, referring to the model's AR, I and MA components, respectively.

2.2 Graph Neural Networks

Scarselli introduced the GNN [17]. The main intuition behind the GNN is to model the data in a graph structure, which allows the GNN to capture non-Euclidean relations.

A graph $G = (V, E)$ is a set of nodes V, and a set of edges E, where $v_i \in V$ represents a node and $e_{i,j} \in E$ describes and edge from node v_i to v_j. The adjacency matrix A describes the edges. A is an $N \times N$ matrix, where N is the number of nodes, with $A_{i,j} = 1$ if $e_{i,j} \in E$ and $A_{i,j} = 0$ if $e_{i,j} \notin E$. A graph can also have node features $X \in \mathbb{R}^{N\times D}$. Here $X_{v_i} \in \mathbb{R}^{1\times D}$ represents the feature vector of node v_i, and D is the total number of features.

Graph convolution is often used to extract information in graph data due to its efficiency. Graph convolution methods fall into two categories: spectral-based methods and spatial-based methods. Spectral-based methods are based on filtering graph Fourier modes or graph frequencies. Spatial-based methods use the graph's structure to update each node's representation. At each convolution, a central node's representation will be updated with the values of its neighbouring nodes.

The diffusion convolutional neural network, introduced by J. Atwood, is an example of a spatial-based graph convolution [18]. Here, graph convolution is treated as a diffusion process. Information is passed between neighbouring nodes by certain transition probabilities, and after a certain number of diffusion steps, an equilibrium in information diffusion is assumed to be reached. The diffusion convolution, as described by J. Atwood, is defined by:

$$H_k = W_k \odot P^k X \tag{4}$$

Here, $P^k = A/rowsum(A) \in \mathbb{R}^{N\times N}$ is the probability transition matrix, $\odot$ is the element-wise product, X is the input feature matrix, H_k is the hidden output of the k^{th} diffusion step and $W_k \in \mathbb{R}^{N\times D}$ is the learnable parameter of the k^{th} diffusion step. The final graph convolution is then given by:

$$Z = \sum_{k=0}^{K} P^k X W_k \tag{5}$$

where $Z \in \mathbb{R}^{\mathbb{N}\times\mathbb{M}}$ is the output signal [19].

Recently, the GNN has shown impressive results in capturing non-Euclidean pairwise correlations. Models combining GNN and Recurrent Neural Network (RNN) techniques have been applied to wind forecasting problems [9], and traffic forecasting [20]. W. Lin has shown that a spatial-temporal GNN, based on the graph WaveNet model, shows great improvement over other forecasting algorithms, such as FNN, RNNs such as long short-term memory (LSTM) or gated recurrent unit (GRU) [15].

However, Lin looked at a limited number of aggregation levels, 1 or 15. It has been shown that deep learning (DL) techniques, including deep neural networks (DNN), convolutional neural networks (CNN), LSTM GRU and bidirectional LSTM/GRU perform differently at different aggregation levels [21]. There has not been an investigation into the performance of a GNN at different aggregation levels.

2.3 Adjacency Matrix Determination

Creating the adjacency matrix is essential for a GNN model to perform optimally. The adjacency matrix can be created using prior knowledge about the nodes. For example, the pairwise geographical distance between the houses can be used to create the adjacency matrix. However, as described in section 1, pairwise non-Euclidean correlations might better capture the relations between houses. A self-learnable adjacency matrix would be preferable as it is difficult to describe the non-Euclidean, low-dimensional relations between houses.

The Graph WaveNet model, proposed by Wu, contains a self-adaptive adjacency matrix to learn latent spatial dependencies between nodes [20]. The self-adaptive adjacency matrix is described as follows:

$$A_{adp} = Q_2(Q_1(E_1 E_2^T)) \tag{6}$$

Here, Q_1 is the Rectified Linear Unit (ReLU) activation function and Q_2 is the SoftMax activation function, and E_1 and E_2 are the source and target nodes, respectively. ReLU is used to eliminate weak connections. SoftMax is used to normalize the self-adaptive adjacency matrix. The node embeddings are learnable and are initialized randomly.

In the context of the GNN, the A_{adp} can be used to represent the transition matrix in equation 5. This means that our graph convolution is now self-adaptive.

3 Methodology

First, section 3.1 will describe the step-wise algorithm used to determine the ARIMA model parameters. Then, in section 3.2 we will describe how the house-based graph structure was represented. After that, we will talk about the two main components of Graph WaveNet, the graph convolution layer (GCL) and the gated temporal convolution network (GTCN), in section 3.3 and section 3.4, respectively. Finally, in section 3.5 we will present the Graph WaveNet model.

3.1 Step-wise algorithm

In their paper, R. Hyndman and Y. Khandakar proposed a step-wise algorithm for forecasting with ARIMA models [22]. The algorithm provides an efficient way to traverse the space of potential models to fit, arriving at the model with the lowest Akaike Information Criteria (AIC) value.

The algorithm first considers four possible models and selects the model with the lowest AIC value. Then, up to thirteen variations of the 'current' model are considered. If a model with a lower AIC value than the 'current' model is found, this model becomes the new 'current' model, and the procedure is repeated. When we cannot find a better model, the process finishes.

3.2 House Based Graph Structure

Usually, STLF problems look at houses separately to predict the electrical load. Spatial-temporal load forecasting models are trained with information on all houses to improve the overall prediction of the model. Therefore, the spatial-temporal electrical load forecasting problem can be seen as a multivariate time-series forecasting problem.

Let $X_i^t = [x_i^t \; x_i^t \cdots x_N^t]$ represent all historical values of features at the t^{th} timestep for N houses. Here $x_i^t = [x_i^{t,1} \; x_i^{t,2} \cdots x_i^{t,D}]$ are D features of the i^{th} house at the t^{th} timestep. Thus the spatial-temporal electrical load data of N houses with D features can be represented as a three-dimensional array: $[X^1 \; X^2 \cdots X^T] \in \mathbb{R}^{N \times T \times D}$.

To embed this data into a graph G, each house is represented as a node, and each edge represents the connectedness between two houses. Thus, $G = (V, E)$, where V is the set of all houses and E is the set of all edges. To efficiently represent E, the adjacency matrix $A_{adp} \in \mathbb{R}^{N \times N}$ is used, as described in section 2.3.

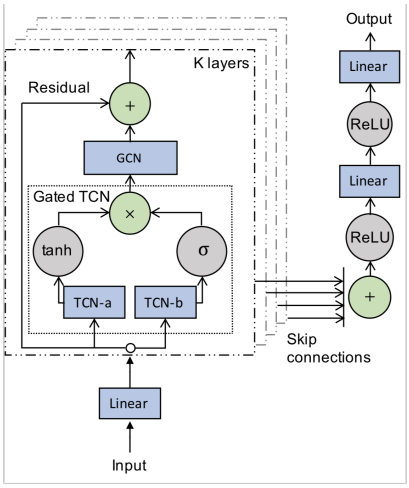


Figure 1: The framework of the Graph WaveNet model [20].

Given the definitions above, we can now formulate the H-step look-back spatial-temporal electric load forecasting problem as learning a function $f : \mathbb{R}^{N \times H \times D} \to \mathbb{R}^N$ that maps historical electrical load data $\{X^{t-H+1}, X^{t-H+2}, \cdots, X^t\}$ to the $(t+1)^{\text{th}}$ timestep. So:

$$f : \{X^{t-H+1}, X^{t-H+2}, \cdots, X^t, G\} \to X^{t+1,d} \quad (7)$$

Here $X^{t+1,d}$ is the d^{th} feature for all houses at the $(t+1)^{\text{th}}$ timestep.

3.3 Graph Convolution Layer

Graph WaveNet proposes a self-adaptive adjacency matrix to learn the latent spatial dependency between nodes with or without prior information [20]. The self-adaptive adjacency matrix A_{adp} is learned end-to-end, using stochastic gradient descent.

By replacing the transition matrix from equation 5 with the A_{adp} as described in section 2.3, the graph convolution can use the diffusion convolution method. This becomes:

$$Z = \sum_{k=0}^{K} A_{adp}^k X W_k \quad (8)$$

3.4 Gated Temporal Convolution Network

We used a GTCN-based model, a dilated causal convolution, to extract temporal features of nodes [23]. We mainly used the dilated causal convolution network because it uses a non-recursive way to handle long-range sequences. This allows for parallel computation. Additionally, this means there will be no gradient explosion.

A gated TCN is used, as controlling the information flow between layers has shown outstanding performance in the past [24]. It allows the model to learn complex temporal features.

3.5 Graph WaveNet

The framework of the Graph WaveNet model can be seen in Figure 1. First, spatial-temporal information is extracted by the GCL and TCN layers. Then, a graph WaveNet framework is used to stack multiple spatial-temporal layers. The framework uses a skip-connection mechanism to handle spatial dependencies at different temporal levels.

The input data is first sent to a gated TCN, followed by a GCL. The layer's output is sent to an external module using skip connections. This module then combines the final output and integrates the individual outputs of the spatial-temporal layers by ReLU activation functions and linear layers. This results in the skip connection mechanism combining the spatial dependencies at different time levels.

Our goal is hour-ahead load forecasting, using only historical hourly electrical load data. So for an output $\hat{X} \in \mathbb{R}^{T_{out} \times N \times D}$, the parameters are $T_{out} = 1$, N is the number of nodes at that aggregation level, and $D = 1$. We will use the mean absolute error (MAE) as the training target of the model.

4 Experimental Setup and Results

This section discusses the experimental setup and result. Firstly, we will go into how the electrical data will be aggregated. Then, we will describe the experimental setup and model parameters. After that, we will go over the evaluation metrics used. Finally, we will present the results.

4.1 Electrical Consumption Aggregation

A dataset that contains the half-hourly energy consumption readings of 5,567 London households that took part in the Low Carbon London project, led by the UK Power Networks between November 2011 and February 2014, was used. [25]. After selecting only the houses with complete readings between 01-01-2013 and 31-12-2013, we were left with 2070 houses. As per previous studies, the hourly readings were used, resulting in a dataset with 8760 entries for all houses [21].

We then created the aggregated data using the hourly data of the original houses $oh = \{1, 2, \ldots, 2070\}$. The oh were randomly selected to create six different aggregation levels $al \in \{3, 10, 30, 100, 300, 2070\}$. Here, the value of al represents the number of oh elements. After that, we generated six groups of average aggregated data ($G3$, $G10$, $G30$, $G100$, $G300$, $G2070$), with a sample size of $gs \in \{690, 207, 69, 20, 6, 1\}$, respectively. The formula used to generate the groups of average aggregated data is:

$$x_m^{al} = \frac{\sum_{n=1}^{al} x_{m,n}}{al} \tag{9}$$

Here, $m \in \{1, 2, ..., gs\}$ is the m^{th} unique subset for aggregation level al, n is the number of aggregated houses and $x_{m,n}$ is the demand of each house in watt-hour per hour (Wh/h).

4.2 Experimental Setup and Model Parameters

As described in section 4.1, the hourly data between 01-01-2013 and 31-12-2013 of 2070 houses in London is used. To create the train, test and validation set, the data is chronologically divided, using a ratio of 0.7 : 0.2 : 0.1, respectively.

Name	Value
Batch size	64
Sequence of dilation factors	(1, 2, 1, 2, 1, 2, 1, 2)
Drop-out rate	0.3
Epoch size	150
Layers of GWN	8
Learning rate	0.001

Table 1: Model parameters

Two variants of the Graph WaveNet model will be tested, self-adaptive Graph WaveNet (Ada-GWN) and non-adaptive Graph WaveNet (nonAda-GWN). Ada-GWN uses the learnable adjacency matrix described in equation 6. The adjacency matrix of nonAda-GWN is the identity matrix. Thus it is not learnable. NonAda-GWN only learns the temporal features without using the graph convolution layer (GCL).

The parameters used to train Ada-GWN can be seen in Table 1. We use eight layers of Graph WaveNet, with a sequence of dilation factors (1, 2, 1, 2, 1, 2, 1, 2). We use equation 8 with a dilation factor $K = 2$. The node embeddings are randomly initialized with a uniform distribution between 0 and 10. We train our model with Adam optimization and a learning rate of 0.001. Eight spatial-temporal layers are used. A drop-out rate of 0.3 is applied to the output of the graph convolution layer. The batch size is 64, and the number of epochs is 150.

We implemented the GWN models using the PyTorch[1] library on a desktop with an NVIDIA Quadro P1000 graphics card, an Intel i7-8750H CPU and 16 GB memory.

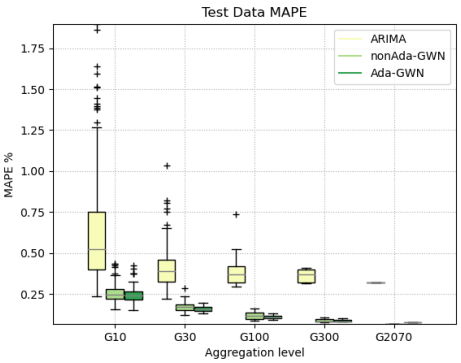
4.3 Evaluation metrics

To evaluate the performance of our models, we will use the mean absolute error (MAE), mean percentage error (MAPE) and the root mean squared error (RMSE). The formulas for these metrics are given in equations 10, 11 and 12.

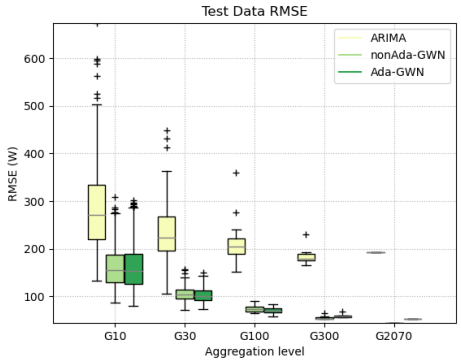
$$MAE = \frac{1}{H} \sum_{j=1}^{H} |\hat{y}_j - y_j| \tag{10}$$

$$MAPE = \frac{1}{H} \sum_{j=1}^{H} |\frac{\hat{y}_j - y_j}{y_i}| \times 100\% \tag{11}$$

[1] https://pytorch.org/



(a) The MAPE errors



(b) The RMSE errors

Figure 2: The MAPE and RMSE errors of the models trained on all aggregation levels.

$$RMSE = \sqrt{\frac{\sum_{j=1}^{H}(y_j - \hat{y}_j)^2}{H}} \tag{12}$$

Here, $\hat{y}_j$ is the predicted value at time j, y_j is the corresponding actual value, and H is the number of time steps.

4.4 Results

In this section, we will go over the results of our experiments. We will first compare the performances of the models on all aggregation levels. Then, the detailed prediction results will be analysed. After that, we will give a comparison of the prediction results of ada-GWN on different aggregation levels. Finally, the training times of the models will be discussed.

4.4.1 Individual load forecasting

The performances of the different models trained on all aggregation levels depicted by the MAPE and RMSE errors on the test data can be seen in Figure 2. The detailed results, including the models' MAE, MAPE and RMSE, can be seen in Table 2. This table shows the models' average performance over all groups at an aggregation level.

We can see that both Ada-GWN and nonAda-GWN perform significantly better than ARIMA at all aggregation levels on all metrics. Furthermore, Ada-GWN slightly outperforms nonAda-GWN at almost every aggregation level. Only at the highest aggregation level, G2070, does nonAda-GWN perform slightly better than Ada-GWN. As there is only one node at this aggregation level, no spatial relationships can be learned, and only temporal information can be used to make predictions. For this reason, nonAda-GWN performs better, as the prediction of Ada-GWN is weakened by its spatial component.

When comparing the performances across different aggregation levels, we can see in Figure 2 that all models are consistent on high aggregation levels. The GWN models are more consistent and accurate on low aggregation levels than ARIMA. From this, we can conclude that the GWN models are better at careful fine-tuning and capturing sensitive data.

The error variance is extremely high at the lowest aggregation level for all models and quickly drops off at G30, where the MAPE improves with an average of 12.4% and the RMSE improves with an average of 54.6 W/group. Table 3 shows the percentage increase between two adjacent aggregation levels. The GWN models show a great improvement in performance on all metrics as the aggregation level increase. ARIMA also shows a good improvement as the aggregation levels increase. However, we see that ARIMA does not benefit as much from the increase in aggregation level as the GWN models.

4.4.2 Detailed prediction results

To better analyse the predictions made by the models, we can look at the detailed prediction results of the 25^{th} and 26^{th} of December in Figure 3. For the analysis, the prediction of the last group at various aggregation levels was chosen, so (G10 - 207) means the 207^{th} group at aggregation level 10.

We can see that at low aggregation levels, the sudden changes in energy consumption make it hard for the models to follow the real demand accurately. Nevertheless, they can follow the main trend of the demand, with the biggest errors being in the peaks. At high aggregation levels, the smoother fluctuations in demand make predicting it easier, resulting in smaller peak-hour errors.

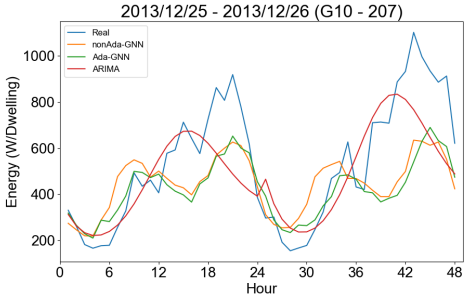
From this Figure, we can also see the difference in the models' abilities to capture complex relations. The GWN models are much better at accurately following the complex real demand, depicted by the blue line. In contrast, ARIMA can only capture the general trend of the demand, resulting in a much more general, simple prediction.

4.4.3 Training time comparison

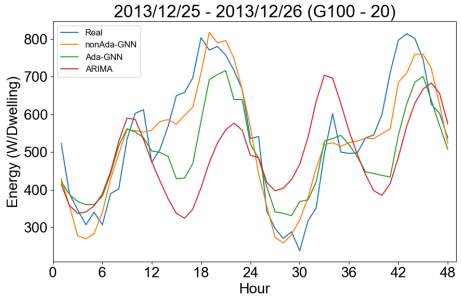
Table 4 shows the average training times per model. Here, the average training time for the GWN models is the average training time per epoch, whereas the average training time for ARIMA is the average training time per dwelling. As these models are trained differently, comparing their training times is difficult. The GWN models can go over all the available data much more efficiently than

Metric	Model	G10	G30	G100	G300	G2070
Test data MAE (%)	Ada GWN	11,36%	7,25%	4,72%	3,92%	3,89%
	ARIMA	23,86%	19,56%	17,38%	15,16%	15,86%
	nonAda GWN	12,02%	8,38%	5,45%	4,13%	3,25%
Test data MAPE (%)	Ada GWN	24,49%	15,50%	10,08%	8,38%	7,89%
	ARIMA	63,26%	43,20%	39,58%	36,25%	31,94%
	nonAda GWN	25,68%	17,54%	11,88%	8,62%	6,60%
Test data RMSE (W)	Ada GWN	159,9	96,5	62,0	51,9	52,5
	ARIMA	290,2	240,6	211,7	186,8	192,9
	nonAda GWN	162,1	111,3	70,6	54,2	44,1

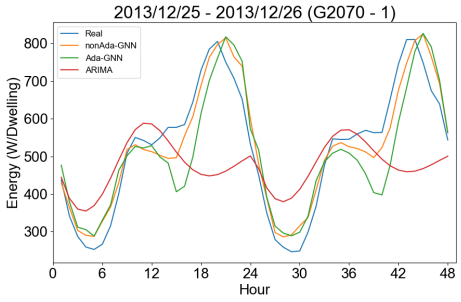
Table 2: Final results of the average performance on all dwellings.



(a) Aggregation level 10



(b) Aggregation level 100



(c) Aggregation level 2070

Figure 3: Detailed prediction results of 25-26 December for different aggregation levels

Model	MAE	MAPE	RMSE
Ada-GWN	22,2%	23,6%	22,63%
ARIMA	9,3%	15,1%	9,4%
nonAda-GWN	27,7%	28,71%	28,15%

Table 3: Average percentage increase between adjacent aggregation levels.

Model	Average training time (s/model)
Ada-GWN	21,5
ARIMA	74,2
nonAda-GWN	17,2

Table 4: Average training time comparison

ARIMA, which can only be trained on one dwelling at a time.

Furthermore, we can see that the nonAda-GWN has a lower training time per epoch than Ada-GWN. This is probably because Ada-GWN takes time to capture the spatial relationships, whereas nonAda-GWN only look at the temporal relations. They are much faster than ARIMA, which takes more than a minute to fit a model into a single dwelling.

5 Responsible Research

This section will first discuss the ethical concepts that were considered. Then we will discuss five principles of the Netherlands Code of Conduct for Research Integrity [26].

5.1 Ethical concepts

Several ethical concepts need to be considered in this paper. Most importantly, as we worked with sensitive personal data, the data must always stay anonymous during the experiments. As the data provided by [25] was anonymous, this was the case.

Secondly, in section 4.1, we decided to leave out almost half of the houses in the dataset. Leaving out such a large amount of data might result in our models not being trained on an accurate representation of the houses. However,

this was done as all houses had to have complete readings between 01-01-2013 and 31-12-2013.

Finally, the results of this paper must be reproducible. To this end, we have published the code used to run the experiments online [2]. The data used to run the experiments can also be found online [3].

5.2 Netherlands Code of Conduct for Research Integrity

In 2018, the Netherlands Code of Conduct for Research Integrity was published to provide guidelines to individual researchers and institutions involved in research [26]. The code presents five principles which should be adhered to when conducting research. We will now review these principles and how they were applied in this paper.

The first principle is honesty. This means the research process is reported accurately and refrains from making unfounded claims or fabricating or falsifying data. All of the claims presented in this paper were found in literature, and all of the presented results were collected through experiments. Furthermore, all the experiments were conducted as presented in section 4.2.

Scrupulousness is the second principle, which states that all methods used are scientific or scholarly. This also applies to this paper, as we used only scientific methods to conduct our research and experiments. Moreover, we objectively presented our results, which means that we did not provide any personal bias or perspective to the discussion in any way.

The next principle is transparency. To this end, we have used publicly available data for our research. Furthermore, we have published the code used to conduct the experiments online to allow anyone to reproduce our results.

Independence is the fourth principle that was taken into account in this paper. As stated previously, this research was presented objectively. There was no influence from non-scientific considerations, such as commercial or political motives.

Lastly, responsibility was considered in this paper. This study was relevant to the problem it is trying to solve. It contributes to a better understanding of the challenges and possible solutions of the short-term load forecasting of electrical demands.

6 Discussion

In this paper, we analysed two variants of the Graph WaveNet (GWN) model, Ada-GWN and nonAda-GWN, for STLF forecasting aggregated electrical demand. The primary objective was to determine the optimal aggregation level for the GWN models, and we compared their performance with the ARIMA model, which served as a benchmark.

Compared to Lin's work, we found that our GWN models exhibited a higher MAPE [15]. One possible explanation is that Lin used data with significantly higher demands, resulting in lower errors.

In the analysis of the ARIMA models, we found that occasionally they were unable to fit a suitable model, resulting in extremely high errors. For this reason, we decided to exclude this data from the analysis, as including it would give an average MAPE of 2.7×10^{16}, and this happened only eight times out of the 303 models fitted in total.

Furthermore, it is important to acknowledge the limitations of this study. Due to time constraints, we used the hyperparameters from previous works, which may not have been optimal for our specific dataset. Conducting a more extensive hyperparameter optimisation process could have potentially improved the performance of our GWN models.

7 Conclusions and Future Work

In this paper, we looked at the performance of short-term load forecasting of the Graph WaveNet model on different aggregation levels by looking at the hour-ahead STLF of a residential electrical demand dataset of 2070 houses in London, Great Brittain. The results showed very low MAPE errors of 6,60% and 7,89% at the highest aggregation level of 2070, comparable to country-level prediction errors. Also, we saw that errors close to 10% could be achieved at an aggregation level of 100.

Besides, although non-adaptive WaveNet outperformed the baseline, self-adaptive Graph WaveNet was consistently and significantly better than non-adaptive Graph WaveNet. This shows that the spatial relationship between houses needs to be adaptively learned to improve the performance of spatial-temporal electric load forecasting.

Furthermore, we saw a consistent improvement in performance as the aggregation levels increased. This indicates that to achieve high STLF performances, we need to look at higher aggregation levels. We also observed a sharp increase in errors at lower aggregation levels, which were discussed in detail. This is a challenge for modern power grids, as there is an increase in decentralized and renewable energy generation sources, which depend on high STLF accuracies to achieve optimal operational performances.

There are also limitations to the Graph WaveNet model. Firstly, the model requires a lot of training data, which cannot have missing values. This will provide challenges if the model is trained and applied to real-life forecasting problems. Secondly, the model needs all the units to have the same length of historical data, which can also be challenging.

[2] https://gitlab.tudelft.nl/lcavalcantesie/flexibily_aggregation_smart_grids

[3] https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london

We recommend seeing if transfer learning can overcome these problems for future work. We would also like to see Graph WaveNet's performance in other forecasting problems, such as industrial electrical load forecasting or wind forecasting.

References

[1] A. Baker, D. Bunn, and E. Farmer, *Load forecasting for scheduling generation on a large interconnected system*. Wiley: Chichester, 1985.

[2] D. Wu, H. Zeng, and B. Boulet, "Neighborhood level network aware electric vehicle charging management with mixed control strategy," *2014 IEEE International Electric Vehicle Conference, IEVC 2014*, pp. 1–6, 03 2015.

[3] A. Małek, M. Kośko, and T. . Łusiak, "Urban logistics of small electric vehicle charged from a photovoltaic carport," *Automobile Archive*, vol. 82, no. 4, p. 63–75, 2018. [Online]. Available: http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-cc995468-03cf-46b9-89c3-a25e69299451

[4] A. Papalexopoulos and T. Hesterberg, "A regression-based approach to short-term system load forecasting," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1535–1547, 1990.

[5] W. R. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-90, no. 2, pp. 900–911, 1971.

[6] C. Tarmanini, N. Sarma, C. Gezegin, and O. Ozgonenel, "Short term load forecasting based on arima and ann approaches," *Energy Reports*, vol. 9, pp. 550–557, 2023, 2022 The 3rd International Conference on Power, Energy and Electrical Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484723000653

[7] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via arma model identification including non-gaussian process considerations," *IEEE Transactions on Power Systems*, vol. 18, no. 2, pp. 673–679, 2003.

[8] C.-M. Huang, C.-J. Huang, and M.-L. Wang, "A particle swarm optimization to identifying the armax model for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1126–1133, 2005.

[9] M. Khodayar and J. Wang, "Spatio-temporal graph deep neural network for short-term wind speed forecasting," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 2, pp. 670–681, 2019.

[10] A. Kavousi-Fard, H. Samet, and F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting," *Expert Systems with Applications*, vol. 41, no. 13, pp. 6047–6056, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417414001912

[11] D. Wu, B. Wang, D. Precup, and B. Boulet, "Boosting based multiple kernel learning and transfer regression for electricity load forecasting," in *Machine Learning and Knowledge Discovery in Databases*, Y. Altun, K. Das, T. Mielikäinen, D. Malerba, J. Stefanowski, J. Read, M. Žitnik, M. Ceci, and S. Džeroski, Eds. Cham: Springer International Publishing, 2017, pp. 39–51.

[12] H. A. Malki, N. B. Karayiannis, and M. Balasubramanian, "Short-term electric power load forecasting using feedforward neural networks," *Expert Systems*, vol. 21, no. 3, pp. 157–167, 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0394.2004.00272.x

[13] A. Taşcıkaraoğlu and B. Sanandaji, "Short-term residential electric load forecasting: A compressive spatio-temporal approach," *Energy and Buildings*, vol. 111, pp. 380–392, 11 2015.

[14] H.-C. Wu and C.-N. Lu, "A data mining approach for spatial modeling in small area load forecast," *IEEE Transactions on Power Systems*, vol. 17, no. 2, pp. 516–521, 2002.

[15] W. Lin, D. Wu, and B. Boulet, "Spatial-temporal residential short-term load forecasting via graph neural networks," *IEEE Transactions on Smart Grid*, vol. 12, pp. 5373–5384, 2021.

[16] F. vom Scheidt, H. Medinová, N. Ludwig, B. Richter, P. Staudt, and C. Weinhardt, "Data analytics in the electricity sector – a quantitative and qualitative literature review," *Energy and AI*, vol. 1, p. 100009, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546820300094

[17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[18] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/390e982518a50e280d8e2b535462ec1f-Paper.pdf

[19] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *CoRR*, vol. abs/1707.01926, 2017. [Online]. Available: http://arxiv.org/abs/1707.01926

[20] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Aug 2019. [Online]. Available: http://dx.doi.org/10.24963/ijcai.2019/264

[21] A. Shaqour, T. Ono, A. Hagishima, and H. Farzaneh, "Electrical demand aggregation effects on the performance of deep learning-based short-term load

forecasting of a residential building," *Energy and AI*, vol. 8, p. 100141, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546822000052

[22] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for r," *Journal of Statistical Software*, vol. 27, no. 3, p. 1–22, 2008. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v027i03

[23] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.07122

[24] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *CoRR*, vol. abs/1612.08083, 2016. [Online]. Available: http://arxiv.org/abs/1612.08083

[25] U. power networks, "Low carbon london project," 2014, [Accessed 07-Jun-2023]. [Online]. Available: https://www.ukpowernetworks.co.uk/

[26] "Netherlands code of conduct for research integrity."