An Exploration of Spatial Temporal Graph Neural Networks for Electricity Load Forecasting

Austin Bell
Columbia University
alb2307@columbia.edu

Ziyin Wang
Columbia University
zw2605@columbia.edu

Malik Drabla
Columbia University
mad2275@columbia.edu

Abstract

This paper explores the use of Spatial Temporal Graph Neural Networks ("STGNN") as a means of node level energy load forecasting across energy grids. While a relatively novel technique, the adaptive ability of STGNNs to learn node level relationships may prove to be extremely beneficial to the energy forecasting needs of grid structures. To this end, we implement a series of STGNN models on the RE-Europe dataset, which consists of 1,494 nodes and their transmission connections plus hourly demand and solar / wind values for mainland Europe across three years. Our results demonstrate that STGCN + Metadata model performed best with a mean absolute error of 6.79. We believe these results demonstrate the effective of spatial temporal graph neural networks for modelling transmission networks due to their ability to capture geographic dependencies when forecasting demand.

1 Introduction

Since the industrial revolution, our earth has been warming due to the release greenhouse gases from burning 'dirty' fuel sources. A recent surge in clean energy investment has led to more wide spread adoption of renewable energy. However some key technical and logistical barriers to the adoption of renewable energy remain:

- Geographically, fossil fuel plants are built remotely and in centralized locations, whereas, renewable energy plants are decentralized and sometimes built in residential areas.
- The large scale storage of renewable energy remains a key challenge, due to the fact that these power plants are non-dispatchable, and their output have limited predictability.

While a wide net of solutions have been proposed and are currently being pursued, our project focuses on the use of neural networks to improve efficiency in the allocation of energy. We propose using spatial-temporal graph neural networks to accurately forecast load for each decentralized node. Through better predictions of energy demand, we can partially address each of these technical barriers.

Traditionally, load forecasting has focused on demand prediction across entire regions due to the confidential nature of energy demand data. Recent efforts to develop datasets with much more fine grained detail have been undertaken. The results are datasets of transmission networks aggregated at the node level. This increased level of granularity enables models to capture geographic effects that may have been masked at the regional or country level.

In our case, spatial analysis enables us to better capture any geographic dependent features related to demand (e.g., nodes providing electricity to cities are likely to be characterized by more extremes). Proximity of nodes in our graphical representation of the energy grid will likely affect demand. As one node reaches capacity they may divert the provision of energy to the nearest node that is below capacity.

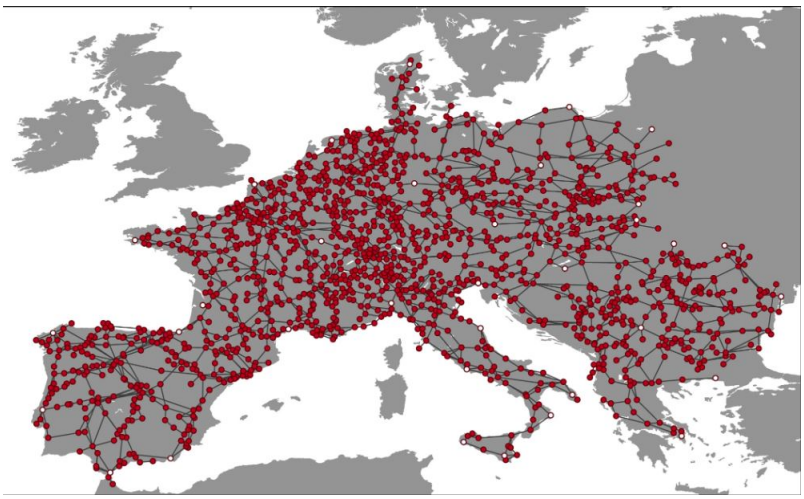

Figure 1: the transmission network in the RE-Europe dataset overlaid on a map of mainland Europe. Each red dot corresponds to a node and each edge corresponds to whether the two nodes are connected. Source: Jensen, T., Pinson, P. RE-Europe, a large-scale dataset for modeling a highly renewable European electricity system. Sci Data 4, 170175 (2017).

Overall, by better understanding our exact energy needs, we will be able to better optimize power flow across decentralized plants, ensure supply more closely meets demand, and thereby, lessen our reliance on advanced storage solutions

2 RE-Europe Dataset

The efforts to generate more granular data for use by researchers to improve our clean energy systems resulted in the RE-Europe dataset [1], which combines a transmission network model, plus data regarding electric generation and demand from 2012-2014. This transmission model identifies relevant nodes and their connections plus transmission capacity. For each hour of the day, demand values are provided for each node. Additionally, solar / wind forecasts are provided every 12 hours and their realized values are provided at the hourly level. All temporal data is provided over a period of three years.

The data covers the entirety of mainland Europe and includes 1,494 unique nodes with corresponding demand. These nodes make up 2,156 connections.

Figure 1 below shows the transmission grid overlaid on a map of mainland Europe. We directly translate this transmission network as the network graph for our graph neural network models.

Initial exploration shows that there is significant variation in load time series data from the RE-Europe Dataset. Figure 2 (left) highlights the variation in demand across three random nodes for a randomly selected 24 hour time period. Figure 2: (right) then highlights the variation of demand across a single node for three randomly selected 24 hour time periods.

3 Relevant Literature

This research combines two areas of ongoing research.

- Machine learning applied to Load Forecasting
- Spatial Temporal Graph Neural Networks

3.1 Machine Learning for Load Forecasting

Prior to the resurgence of deep learning, most load prediction models utilized more traditional statistical algorithms. Within this group of statistical algorithms, there has been research utilizing

linear regression [2], multiple regression models [3,4], Autoregressive moving average (ARMA) models [5], and autoregressive integrated moving average (ARIMA) models [6,7].

More recently however, there have been a variety of neural network approaches that look to capture the temporal features of load forecasting. The most popular approach is using RNN's with add-ons or adjustments to model day ahead or week ahead demand. Polson and Sokolov (2018) create a new loss function leveraging extreme value theory to better model extreme peaks and troughs of energy demand[8]. Bouktif et al (2018) leverages a genetic algorithm to identify the optimal LSTM and feature configuration for load forecasting [9]. Additional networks have been implemented including deep belief networks (Ouyang et al 2017) [10] and more complex architectures combining various convolution and RNN layers (He 2017) [11].

To date, our team has only been able to find one model which utilizes the same RE-Europe data. Zhu et al (2020) [12] develop a seq2seq model combined with a multiscale network and multitask learning. The multi task approach minimizes the combined loss of the seq2seq and multiscale models. For reasons discussed in Section VI: "Experimental Results", we do not include their results in our comparisons due to an inability replicate. But we want to make clear that this is not due to their modelling approach, but instead due to their data normalization and that we are unclear what they are actually predicting.

In each of these, we note that similar features are used as inputs to the model including weather features (forecasted temperature, wind, sun), current and previous load series, and temporal identifiers (holiday, day of week, season, etc.).

3.2 Spatial Temporal Graph Neural Networks

On the other side, significant ongoing research is being conducted on spatial temporal graph neural networks, but we focus on advances within traffic forecasting and passenger demand prediction. Originally developed by Bing Yu and his team in 2018 [13], multiple improvements have occurred over the past two years. Their underlying model consists of multiple spatial-temporal blocks, which are each made up of three convolution layers (temporal convolution + graph convolution + temporal convolution). Their model built upon Dauphin et al (2015) after they introduced Gated Linear Units (GLU) [14].

Various improvements have since expanded on the original STGCNs. Guo et al (2019) [15] added an attention mechanism to better attend to road networks with more influence. This attention mechanism is built off of Velickovic (2018) work on Graph Attention Networks [16]. Bai et al (2019) [17] combines the spatial temporal graph with a seq2seq architecture with attention for their predictions. Wu et al (2019) [18] utilizes both a graph convolution and a causal convolution. Finally, Song et al (2020) [19] first develops a localized spatial temporal graph by connecting each node with itself in the adjacent time steps. These localized graphs along with learnable temporal and spatial embeddings serve as input into the STGNN.

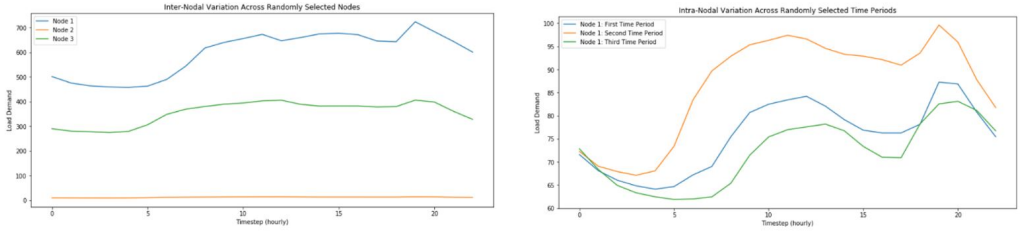

Figure 2: (left): Electricity Load for a 24-hour time period plotted for three randomly selected nodes. While a similar pattern is present across all three nodes for the 24 hour time period, there is significant variation in energy demand for each node.
Figure 2 (right) : Electricity Load for the same node across three randomly selected 24 hour time periods highlights the variation that can occur within a single random node of our data.

4 Methodology

A spatial temporal graph neural network is a small expansion on graph convolution networks. Where, traditionally, graph neural networks only consisted of a spatial component (i.e., graph convolutions on an adjacency matrix), a STGNN adds a temporal component. Currently, temporal models include RNNs, Transformers, as well as temporal convolutions. As of now, temporal convolutions are typically used in STGNNs as demonstrated in Yu et al's spatial temporal graph convolution network due to their speed.

In our research, we explore four separate spatial temporal graph neural networks. Each of these are composed of at least two of three building blocks. Below, we derive and explain how each of these building blocks function.

4.1 Graph Convolution Network

Originally implemented Kipf and Welling (2016) [20], graph convolution networks consists of layers of graph convolutions applied to the normalized adjacency matrix in combination with our input features. We can then define our graph convolutions within our network as the following:

$$H^{l+1} = \sigma(A_{norm} H^l W^l) \tag{1}$$

Where H^l is layer output at layer (l), W is our trainable weights matrix at layer (l), A_{norm} is our normalized adjacency matrix with self loops.

Further, this can be reduced leveraging the same methodology as Kipf and Welling. First, we define a spectral graph convolution as

$$g_\theta * x = U g_\theta U \tag{2}$$

where $g_\theta * x$ is our graph convolution (i.e., a function of eigenvalues of our graph Laplacian) parameterized by $\theta \epsilon R^N$ in the fourier domain. U are the matrices of eigenvectors of our normalized graph Laplacian $L = I - A_{norm} = U g_\theta U$, where I is a identity matrix

This equation is very computationally expensive. Therefore Kipf and Welling further reduce this to less computationally expensive equation of:

$$g_\theta * x \approx \theta(I + A_{norm})x \tag{3}$$

This final equation is used within our network.

4.2 Temporal Convolutions

Typically recurrent neural networks have been used to model sequences, but recent literature has demonstrated that 1-D causal convolutions with kernel width (K_t) have equivalent effectiveness, yet train faster and have no dependency constraints. [21] Yu et al (2018) utilize temporal convolutions rather than recurrent neural network for their STGCN to capture the temporal dynamics of traffic flows.

For each node in the graph, the convolution explores Kt neighbors of the input features without any padding. This shortens the dimension of our output by K_{t-1} each time. The 1-D causal convolution is follow by a gated linear unit (GLU) as a nonlinearity. The input features X for each node is 2-D tensor where $X \epsilon\ R^{t \times f}$, where t is the number of timesteps and f is the number of features (or channels). The convolution kernel then maps the input X to a single output element. The temporal convolution can then be defined as:

$$\Gamma * X = P \odot \sigma(Q) \tag{1}$$

Where $\odot$ is the element wise Hadamard product and P and Q are separated 1-D convolutions with kernel width Kt on input X. P and Q act as input gates within the GLU, where $\sigma(Q)$ determines which time series features to pass on. Additionally, for stacked temporal convolutions we add residual connections.

4.3 LSTM

A recurrent neural network models an input sequence $[x_1, x_2, ..., x_n]$ in order, such that:

$$h_t = f_t(h_{t-1}, x_t) \tag{1}$$

Where h_t is our hidden layer at timestep (t) and is the vector representation of the sequence up until this point. x_t is the corresponding data input at timestep (t). This is repeated until all inputs of the sequence have been included. The original formulation of the recurrent network led to vanishing / exploding gradients, therefore, Hochreiter and Schmihuber (1997) [22] developed the LSTM Cell.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t * \tanh(C_t) \tag{7}$$

The purpose of the cell is to learn what information to pass on as the recurrent neural network progresses across the sequence. Therefore, the cell is applied at each timestep. We denote i as the input gate, f as the forget gate, and o as the output gate. C and $\tilde{C}$ represent the cell state.

5 Our Approach

We use these spatial and temporal models at each time sequence (e.g., 24 hour historical load and solar data) to predict the next 24 hours. We can define our problem via a series of temporal graphs ($G_t = (V_t, E, W_t)$), where G_t refers to the graph at timestep (t), V_t refers to the vertex at timestep (t), E refers to the edges, and W_t corresponds to each nodes' weights at timestep (t)

In our model, the graph is represented by the transmission network (figure 1). Therefore, we represent each node in the graph by a node bus that delivers electricity for the population. Two nodes are connected in the graph if they are connected by a transmission line in the real world.

For each model, we train 200 epochs with a starting learning rate of .001 that halves after every 50 epochs. We optimize using the Adam algorithm and minimize based on mean squared error. Our research explores four separate models, which are detailed below.

5.1 Data Processing

First, we find that eight nodes within the data have a historical load of 0 for all hours across all three years, therefore, we remove these nodes. Afterwards, there are 1,487 nodes in total.

As noted before, there is significant variation within our nodes, therefore, we need to normalize each of our inputs. We use min max scaling to scale all of our nodes values between 0 and 1. We normalize each node independently of one another such that the variation within each node will be between 0 and 1. We use the following formula for our normalization.

$$x^{ij}_{norm} = \frac{x^{ij} - min(x^i)}{max(x^i) - min(x^i)} \tag{1}$$

Where x^i is our data for node (i) and x^{ij} is the demand of node (i) at timestep (j).

Our goal is to use the previous day's historical time series data (hours 00:00 through 23:00) to predict the next day's energy demand (hours 00:00 through 23:00). Since we utilize a graph approach, we input all nodes at a time to the model. The input of model, excluding batch size, is a 3-dimensional tensor $X^{n \times t \times f}$, where n is the number of nodes, t is the number of timesteps, and f is the number of features. As denoted, there are 1,487 nodes, 24 timesteps (one per hour), and we use two temporal features. We use the normalized load demand for each node and the solar levels as a proxy for weather.

In order to take advantage of the spatial interconnectedness of our transmission model, we must use our graph's adjacency matrix (A). However, according to Kipf and Welling, a normalized adjacency matrix with self-loops is required, therefore, we preprocess our graph accordingly.

normalize our adjacency matrix, where D is our graph's degree matrix:

$$A_{norm} = D^{-1/2} A D^{-1/2} \tag{2}$$

We then add self-loops which guarantees that the current node is included in the summation of the neighborhood.

$$A_{norm} = A_{norm} + diag(A_{norm}) \tag{3}$$

5.2 STGCN

A spatial temporal graph convolution network ("STGCN") is composed at a minimum of two STGCN blocks, a final temporal convolution, and a final fully connected layer. Each STGCN blocks is composed of a graph convolution sandwiched between two temporal convolutions. This network is outlined in detail in figure 3.

5.3 STG-LSTM

Our second model is a spatial temporal graph convolution model, but we replace the temporal convolution with a bidirectional LSTM. In each LSTM layer, there exists a single LSTM that is applied to each. The outputs of each layer in the Bi-LSTM are averaged. Figure 4 below displays a diagram of this network.

We also experimented with attention layers, but found no improvement.

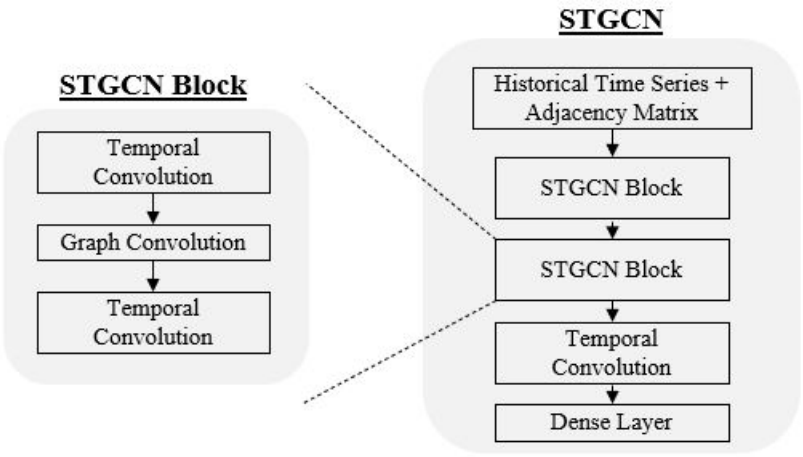

Figure 3: a spatial temporal graph convolution architecture originally developed (Yu et al 2018). It takes as input our historical time series and normalized graph adjacency matrix. This passes through two STGCN blocks, a final temporal convolution, and a fully connected layer. Each STGCN block is composed of a temporal convolution + graph convolution + temporal convolution.

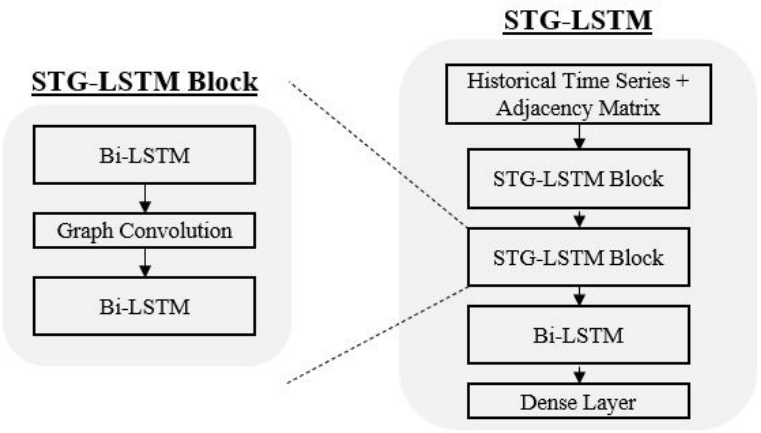

Figure 4: A spatial temporal graph convolution where the temporal convolutions are replaced with bidirectional LSTMs. The output of each layer in the Bi-LSTM are averaged. A single LSTM is used at each layer and applied to each node individually – the results of this loop are concatenated.

5.4 Incorporating Metadata

In the previous models, we only utilized historical solar and load data to predict the next day's energy demand. However, this leaves out a significant amount of information that may be useful in forecasting. For these two models, we supplement our network with additional static features based on the node and date that is being forecasted.

Prior to training our network, we extract the mean and variance for each node's demand by day of the week and by season. For example, one entry in this lookup table would be node 1, Thursday, Winter. Additionally, we develop a lookup table that contains the holidays for each country. Then depending on the day that is currently being forecasted, we can extract a vector that includes: mean load of node / day / season, average load variance of node / day / season, and a binary indicator for whether it is a holiday.

Figure 5 demonstrates how this is incorporated into our model.

Our hypothesis is that the linear layer following the day-ahead summary statistics will essentially predict the average of the day being forecasted. The STGNN is then only responsible for predicting the variation for each timestep. The result is that the STGNN simplifies and predictions improve.

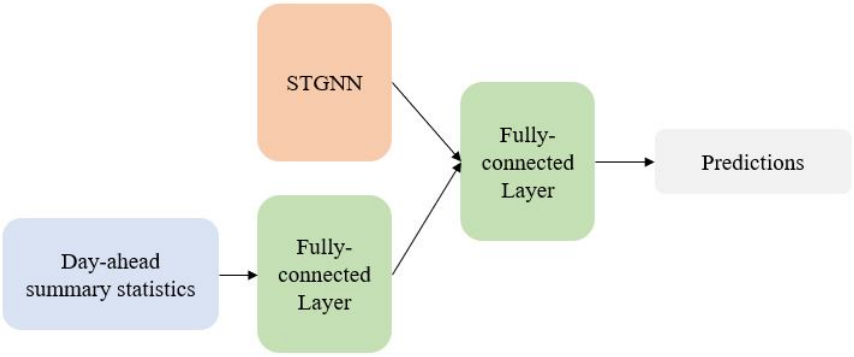

Figure 5: The architecture combining our STGNN models with the day ahead summary statistics. The day ahead summary statistics include : mean of node / day / season, average variance of node / day / season, and a binary indicator for whether it is a holiday.

6 Experimental Results

Table 1 below displays the results across three models. We used the last three months of our data (October 1st, 2014 through December 31st, 2014) as our validation set. While, we normalized our data for training, we have de-normalized our predictions for evaluation. We believe this should make our results more comparable with future research.

We evaluate our model on three separate measures: Mean Squared Error, Mean Absolute Error, and Root Mean Squared Error. [1]

	MAE	**MSE**	**RMSE**
STGCN	8.280	434.518	20.845
STGCN + Metadata	**6.789**	**323.124**	**17.976**
STG-LSTM + Metadata	8.016	410.630	20.264

Table 1: Model results across three STGNN based models. We present de-normalized results across three measures including mean squared error, mean absolute error, and root mean squared error.

Our results demonstrate that the traditional STGCN which incorporates metadata has the best results. Further, the figures below display the results across our models. We include three figures evaluating different levels of granularity of our model predictions.

Figure 6 (top left) displays the prediction for the first node and the first prediction (October 2nd, 2014 00:00-23:00). The second chart, Figure 6 (top right) displays the averaged predictions for the first node across the entire validation time period. Finally, the last chart, Figure 6 (bottom), displays the averaged predictions across all time-periods and across all nodes. As prediction granularity decreases, our model performance improves.

7 Conclusion

In conclusion, we find that utilizing spatial temporal graph neural networks for the purpose of more granular load forecasting is a promising approach. Traditionally, load forecasting was completed using solely recurrent networks, which does not capture any of the spatial features that might be present in a real-world transmission model.

Our team also believes that this graph neural network approach for representing transmission networks will have many more applications in the future. One area that we want to highlight is the combination of reinforcement learning and graph neural networks for the optimization of allocation of energy. Renewable energy is produced in a very decentralized format, can take time to generate, and is costly to store or lose. By simulating various energy generation, demand, and supply scenarios, engineers can train reinforcement learning models to learn how best to allocate energy given a

[1] Earlier we noted that we do not include the results from Zhu et al (2020) in our comparison because they do not de-normalize their prediction results back to the original load data. Therefore, it is a meaningless comparison if we do not normalize our data in the exact same way. Based on our understanding of their research paper, we do normalize our data in the same way, but it is our belief that they did not actually normalize each node independently, but rather they mistakenly normalized all nodes together. Some nodes have a range with 0 and 10, whereas, some nodes have a range in the thousands. Therefore, normalizing nodes jointly leads to nodes with high values equalling approximately a constant 1 and nodes with low values equalling a constant 0.

We believe that this is confirmed by their presented results. Their best performing model resulted in a mean squared error of $4.9210e{-}7$ (rewritten as 0.000000492). If a node has a variation somewhere between 0 and 1 or even .25 and .75, then this would suggest a near perfect prediction. If the reader would engage in a small thought experiment: assuming that the error is constant this would assume a mean absolute error of 0.0007, so on average if the target value was 0.35 for a particular time step then the prediction would be .3507. We found this to be an unlikely result. However, what is in fact happening is that because of the massive variation between nodes, if the researcher jointly normalizes the load data then a significant number of nodes would have near constant normalized load as described in the paragraph above. Therefore, a model that predicts this near constant value for each node would generate the above predictions. This model would be seemingly performing well without predicting any of the actual variation in energy demand. We were able to replicate similar results by jointly normalizing our node data and running our models.

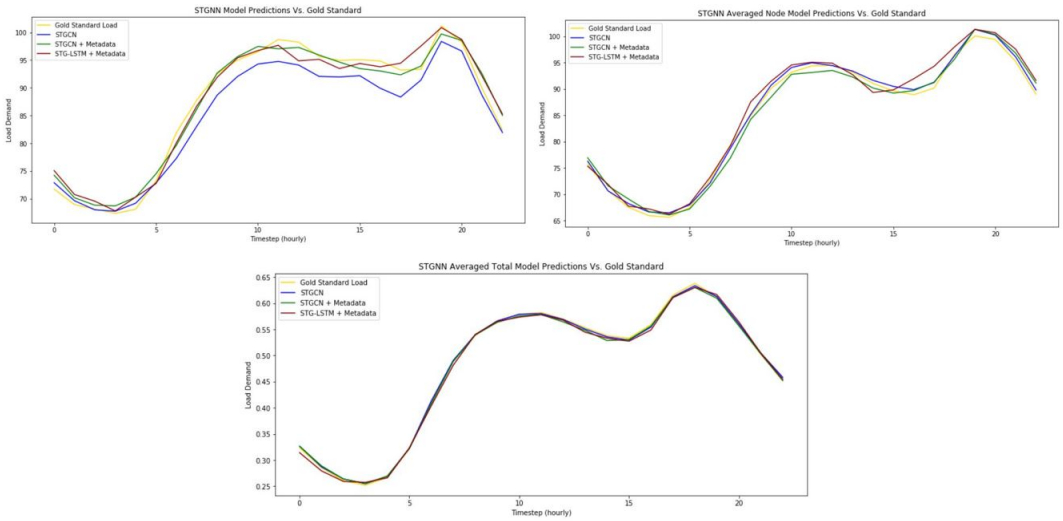

Figure 6: (top left): displays the prediction for the first node and the first 24-hour prediction (October 2nd, 2014 00:00-23:00) for load demand. Figure 6 (top right): displays the averaged predictions for the first node across the entire validation time period. Figure 6 (bottom): displays the averaged predictions across all time-periods and across all nodes

variety of constraints (e.g., travel time, transportation costs, transmission capacity, etc.). Models that significantly improve how energy is allocated across will have lasting real world impact.

Acknowledgments

The authors would like to thank Tue V. Jensen and Pierre Pinson at the Technical University of Denmark whose efforts obtained the RE-Europe dataset. We would also like to thank Professor Drori who guided the initial efforts of this project.

References

[1] Jensen, T., Pinson, P. RE-Europe, a large-scale dataset for modeling a highly renewable European electricity system. Sci Data 4, 170175 2017.

[2] A. Goia, C. May, G. Fusai, Functional clustering and linear regression for peak load forecasting, Int J Forecast, 26 (4) (2010), pp. 700-711

[3] R. Ramanathan, R. Engle, C.W. Granger, F. Vahid-Araghi, C. Brace, "Short-run forecasts of electricity loads and peaks", Int J Forecast, 13 (2) (1997), pp. 161-174

[4] Close Amral N, Ozveren C, King D. Short term load forecasting using multiple linear regression. In: UPEC 2007. 42nd international universities power engineering conference, 2007, 2007. p. 1192–8

[5] S. Pappas, L. Ekonomou, D. Karamousantas, G. Chatzarakis, S. Katsikas, P. Liatsis, "Electricity demand loads modeling using autoregressive moving average (ARMA) models", Energy, 33 (9) (2008), pp. 1353-1360

[6] C.-M. Lee, C.-N. Ko, "Short-term load forecasting using lifting scheme and ARIMA models", Expert Syst Appl, 38 (5) (2011), pp. 5902-5911

[7] Cho M, Hwang J, Chen C-S. "Customer short term load forecasting by using ARIMA transfer function model." In: Proceedings of EMPD '95., 1995 international conference on energy management and power delivery, 1995. vol. 1; 1995. p. 317–22

[8] Michael Polson and Vadim Sokolov, Deep Learning for Energy Markets, Arxiv, 2018

[9] Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani, Optimal Deep Learning LSTM Model for Electric

[10] Tinghui Ouyang, Yusen He, Huajin Li, Zhiyu Sun, and Stephen Baek, A Deep Learning Framework for Short-term Power Load Forecasting, Arxiv, 2017

[11] Wan He, Load Forecasting via Deep Neural Networks, Procedia Computer Science, Volume 122, 2017, Pages 308-314

[12] Zhu H, Zhu Y, Wang H, et al. Multi scale deep network based multistep prediction of high-dimensional time series from power transmission systems. Trans Emerging Tel Tech. 2020; e3890. https://doi.org/10.1002/ett.3890

[13] Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In IJCAI.

[14] Dauphin, Yann; Fan, Angela; Auli, Michael; and Grangier, David. (2016). Language Modeling with Gated Convolutional Networks.

[15] Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019a. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 922–929.

[16] Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In International Conference on Learning Representations

[17] Bai, L.; Yao, L.; Kanhere, S.; Wang, X.; and Sheng, Q. 2019. Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In IJCAI

[18] Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph wavenet for deep spatial-temporal graph modeling. In IJCAI

[19] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan, 2020, Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting, AAAI.

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

[21] Gehring, J., Auli, M., Grangier, D., Yarats, D. Dauphin, Y.N.. (2017). Convolutional Sequence to Sequence Learning. Proceedings of the 34th International Conference on Machine Learning, in PMLR 70:1243-1252

[22] Hochreiter, Sepp Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.