Deep Learning Milestone Paper

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 (STGNNs) 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 standard STGNN on the RE-Europe dataset, which consists of 1,494 nodes and their transmission connections plus hourly demand and solar / wind values and solar / wind forecasts every 12 hours. Initial results from our STGNN model showed a mean squared error loss of 6.62e-5. We believe this less than stellar result was due in part to the graph convolution's compression of valuable node information within neighborhoods.

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 widespread 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.

Spatial analysis enables us to better capture any geographic dependent features related 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.

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, lessening our reliance on advanced storage solutions

2 Methodology

The goal of our project is to improve forecast prediction of load demand for each node (i.e. power provider) through leveraging our implementation of a Spatial Temporal Graph Neural Network

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

(STGNN). Our power provider nodes will represent our graph nodes and their transmission connections will represent the edges. Each node includes a variety of data at each timestamp such as hourly demand, wind / solar forecasts, and wind / solar realizations.

2.1 Graph Convolution Network

Originally implemented Kipf and Welling (2016) [1], 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:

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

2.2 Spatial Temporal Graph Convolution Networks

A spatial temporal graph convolution network ("STGCN") [2] is a small expansion on the original graph convolution networks. Where, traditionally, graph neural networks only consisted of a spatial component (i.e., graph convolutions on an adjacency matrix), a STGCN adds a temporal component. Currently, temporal models include RNNs, Transformers, as well as temporal convolutions. As of now, temporal convolutions are typically used in STGCNs due to their speed. We use these spatial and temporal convolutions at each time sequence (e.g., 24 hour historical load and solar data) to effectively forecast our time series.

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).

A STGCN is then composed of two STGCN blocks and a final temporal convolution. Each STGCN block consists of a temporal convolution, a graph convolution, and another temporal convolution.

Each pass of the network is applied to each G_t. Further explanations of the STGCN and STGCN blocks can be found in algorithm boxes 1 and 2, respectively.

Algorithm 1: STGCN Network Training

Result: Trained STGCN Network
Inputs: $A_{norm}, \boldsymbol{X}$;
while $i < Num\_Epochs$ **do**
  while $b < Num\_Batches$ **do**
    $H^1 = STGCN\_Block(A_{norm}, \boldsymbol{X_b})$;
    $H^2 = STGCN\_Block(A_{norm}, \boldsymbol{X_b})$;
    $H^{out} = Temporal\_convolution(\boldsymbol{X_b})$;
    $b = b + 1$;
  end
  $i = i + 1$;
end

Algorithm 2: STGCN_Block

Result: Hidden Layer from a STGCN Block
Inputs: $A_{norm}, \boldsymbol{X}$;
$H^1 = Temporal\_convolution(\boldsymbol{X_b})$;
$H^2 = Graph\_convolution(A_{norm}, \boldsymbol{X_b})$;
$H^{out} = Temporal\_convolution(\boldsymbol{X_b})$;

2.3 Evaluation

Experiments are run to forecast day ahead demand. For each experiment, we evaluate the time series output using mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean squared error (RMSE).

3 RE-Europe Dataset

We utilize the recently developed RE-Europe dataset which consists of nodal load demand across mainland Europe and also includes a transmission network model, plus data regarding electric generation and demand from 2012-2014.[3] There are 1,494 different nodes with 2,156 transmission connections. Typically, load demand at the node level is highly confidential data, therefore, there are no other datasets including this granular of information.

In the below figures, we display some of the variation seen across our nodes' load demand. Here we display two types of variation: Figure 1 showcases variation across load demand levels for three randomly selected nodes for the same 24 hour time periods and Figure 2 highlights variation within a single node across three randomly selected 24 hour time periods.

4 Summary of Progress

Our team has made significant progress in our project so far. The first milestone was aimed at setting our baseline and implementing a (hopefully) near state of the art solution. Whereas, the following half of the project will be focused on implementing and expanding upon current state of the art techniques within STGNNs. Our achievments thus far include:

- Processed, standardized, and prepared the data for spatial temporal forecasting
- Developed a DGL Graph Convolution Network that ignores the longitudinal component
- Implemented the baseline Spatial Temporal Graph Convolution Network

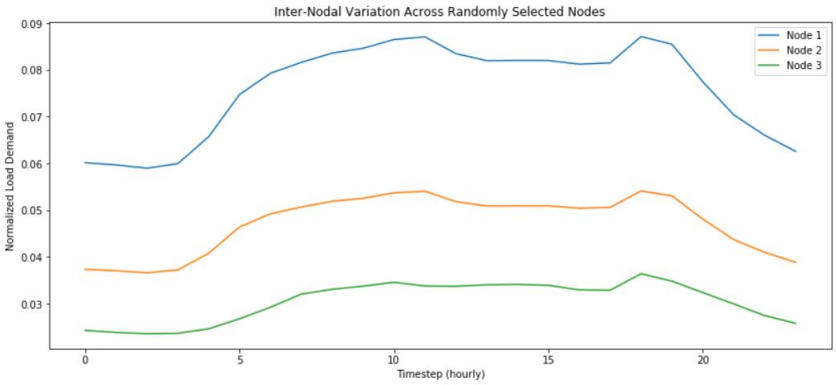

Figure 1: Three randomly selected nodes are selected and plotted for the same set of 24 hours demonstrating the variation in load demand across our nodes

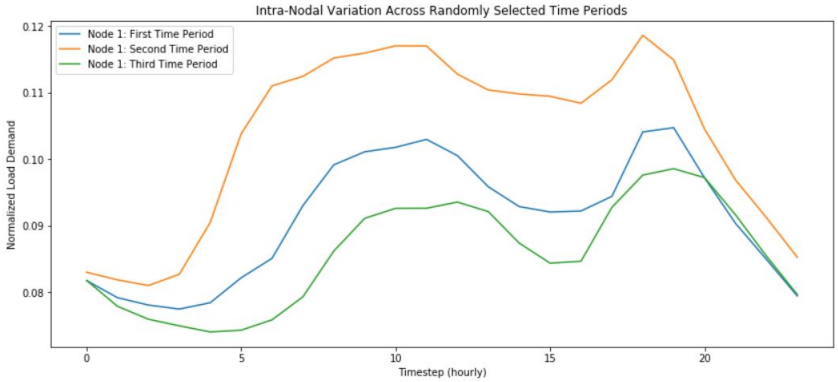

Figure 2: Three randomly selected sets of 24 hour time periods across a single randomly selected node demonstrating the variation in load demand within a single node

4.1 Data Processing

Our first methodology involved taking the provided data and reformatting it to work with our spatial temporal GNN. Our initial baseline model just takes a graph and a tensor of our longitudinal features. Input features are represented by a 3D tensor $\boldsymbol{X}^{n \times t \times f}$, where n refers to our number of nodes, t responds to the number of historical timesteps, and f refers to our number of longitudinal features.

Prior to any processing, we focused general feature engineering given our original data. This included identifying the day of the week (whether it was a weekend), identifying the month, identifying the season, and identifying whether or not a holiday occurred on that date. Figuring out whether a holiday occurred required slightly more work as our data is based across all of Europe, where each country has their own set of holidays.

There are a significant number of preprocessing methodologies that we could pursue, therefore initially, we focused on developing robust enough data processing code that enables us to rapidly experiment with a variety of input data. This includes using different methods to normalize our data, utilizing a different number of historical timesteps as our input, and including different sets of features in our model. We currently normalize our load demand between 0 and 1 with the following formula:

$$x_{norm}^{ij} = \frac{x^{ij} - min(x^i)}{max(x^i) - min(x^i)} \tag{4}$$

Finally, our baseline model is written in pure Pytorch and excludes Pytorch Geometric. This means that we are unable to pass a graph into the network and must instead utilize our graph's adjacency matrix. In order to effectively utilize our adjacency matrix, two processing steps were required:

1. Normalizing it with the formula

$$A_{norm} = D^{-1/2} A D^{-1/2} \tag{5}$$

   where A is our adjacency matrix and D is our degree matrix

2. Adding self-loops to ensure that we include the current node and not just those in the neighbourhood. self loops are defined as:

$$A_{norm} = A_{norm} + diag(A_{norm}) \tag{6}$$

4.2 DGL Graph Convolution Network

First of all, we use DGL library to build our GCN model. In this model, we use a basic graph convolution network architecture. Our goal with DGL was to determine whether any features were predictive of load irrespective of the temporal component. Therefore, we split our data and generate one graph for each time stamp. For each timestamp, we predict load using all other features.

For GCN model, we train for 100 epochs(with allowed early stopping), at a learning rate of 0.01 that decrease every 5 epochs if the model does not improve.

4.3 Spatial Temporal Graph Convolution

As mentioned our baseline model is based on the original STGCN.

For the baseline, we train for up to 200 epochs (with allowed early stopping) at a learning rate of .001 that decreases every 50 epochs if the model is not improving.

5 Evaluation

Our team evaluated across all available nodes in our dataset on the last three months in our data (October 1 st , 2014 through December 31 st , 2014).

5.1 DGL

As can be seen in Figure 3, validation MSE and training MSE do not decrease after 20 epochs and finalizes with very poor performance. This is unsurprising, because while our other features may assist in predicting load they are not the key predictors. This confirms our hypothesis that a temporal component is needed to predict demand and historical load information, must also be used.

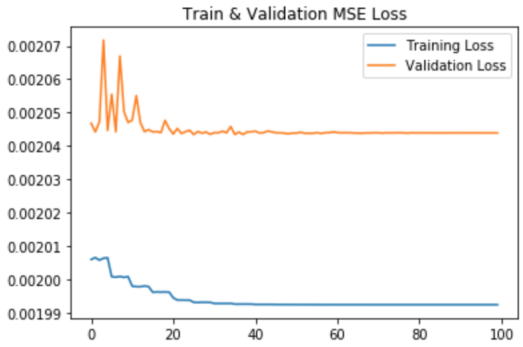

Figure 3: Training MSE VS Validation MSE

5.2 STGCN

Table 1 compares our model across three metrics to the current state of the art using the RE-Europe dataset for load demand forecasting. We note that this is also the only paper that we have identified leveraging this dataset for spatial temporal prediction. Their model utilizes a multi-scale network combined with a seq2seq network.[3]

	MAE	MSE	RMSE
STGCN	.0038	6.62e-5	8.1e-3
Multi-Scale + Seq2Seq	N/A	4.92e-7	N/A

Table 1: Preliminary results comparing our Baseline STGCN to current state of the art Multi-Scale Seq2Seq architecture for RE-Europe dataset. We measure our model across three common measures: mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). Only MSE is reported for Multi-Scale Seq2Seq architecture.

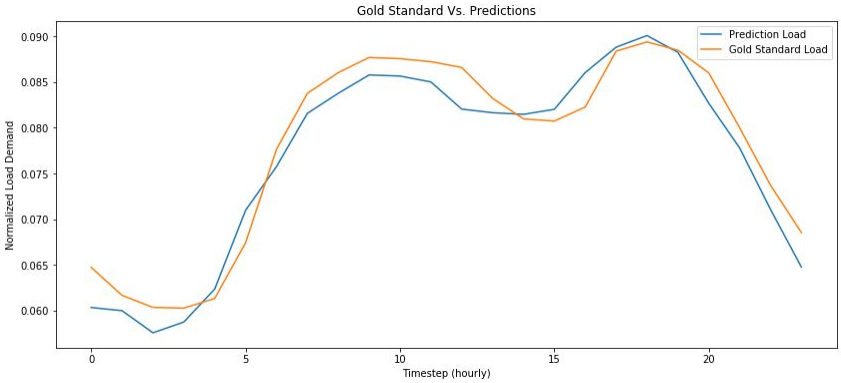

Figure 4: The above figure compares the gold-standard load demand values to the STGCN predicted values. The selected 24 hour time period and node were chosen due to the **low error** between the true load demand and STGCN predicted values

Figures 4 and 5 below display two selected predictions from our STGCN. Figure 4 selected a node and 24 hour time period that resulted in low error. Whereas, Figure 5 selected a node with high error between the predicted values and the gold standard.

6 Discussion

Our current hypothesis for these poor results is that STGCNs are typically used for forecasting traffic, which does not have the same level of temporal dependencies as day ahead load forecasting. Typically, load forecasting requires much longer historical timestep inputs and must also predict much longer forecasts. Therefore, we believe that the basic temporal convolution is not sufficient for our problem.

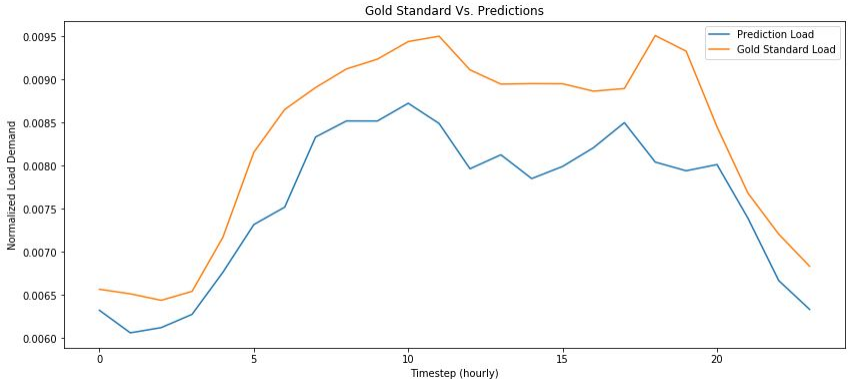

Figure 5: The above figure compares the gold-standard load demand values to the STGCN predicted values. The selected 24 hour time period and node were chosen due to the **high error** between the true load demand and STGCN predicted values

Therefore, our next step is to implement a spatial temporal graph neural network that includes a sequence model for the temporal component such as the one developed by Bai et al (2019) [4] . Their team combines the spatial temporal graph with a seq2seq architecture with attention to better model passenger demand for ride hailing services in China. We believe that this seq2seq architecture will rapidly improve our models' effectiveness.

Additionally, we believe that our current model is missing two key pieces of available data. The first is that we are excluding our solar and wind forecasts since we only include historical data at the moment. However, we expect that including weather indicators would signify whether people stay indoors and use more energy or go outdoors. The second piece is that we are excluding the historical and current day metadata. This includes whether the day is a holiday, which season, etc. We excluded this because it decreased our results when including as a traditional time series. However, past research shows that by incorporating metadata as cross-sectional data via a linear layer leads to improved results.

Finally, based on our recent experiments, it appears that tuning our hyper-parameters will be quite important. First, small changes in the learning rate has had big impacts on our model's outcome and finding the optimal rate has proved elusive. Our team will work on tuning the learning rate while also implementing a more sophisticated learning rate scheduler. Secondly, we trained our model for 200 epochs and the model was still improving slowly. The current state of the art trained for around 800 epochs. We will ultimately train for this long, but will wait until we have a more complete model.

7 Conclusion

Despite the poor original results, we are quite happy with the progress that our team has made so far. We have fleshed out our methodology, processed the data, and fully implemented then evaluated our two baseline models. Additionally, since this is a longer term project, we have implemented a robust framework that will now allow for rapid and efficient deep learning experimenting. We plan to efficiently use the next month to explore and, subsequently, surpass the current state of the art.

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

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

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

[4] 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

[5] 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