Utilizing Neural Networks to Predict Electricity Demand

Mark Barna
Data Science Capstone

Background

The electricity grid is currently experiencing a rapid transformation from fossil fuel-based generation to cleaner energy sources. In 1990, solar and wind power generated just 0.1% of all electricity in the United States [1]. By 2007, that number had increased but only to 0.8%. However, in the next ten years to 2017, solar and wind generation in the United States jumped to 8.2%. The intermittent nature of wind and solar generation introduces new challenges in managing the grid to balance supply with demand. One important aspect of grid management, particularly when incorporating renewables, is predicting electricity demand.

In recent years, researchers have proposed new methods to apply neural networks to projecting electricity demand. Chen et al. used the "extreme learning machine" technique to speed the network learning process and avoid stopping in local optima [2]. He et al. employed a chaos theory technique known as the Chebyshev map to reduce overfitting [3]. In addition, novel techniques for modeling weather forecasts show promise in helping to predict electricity demand since energy usage is often weather dependent [4].

Further, recurrent neural networks, especially those using the long short-term memory (LSTM) architecture, have been used effectively to analyze time series data [5]. Unlike feedforward neural networks (FFNs), which seek to map inputs to outputs irrespective of sample order, LSTMs can learn patterns in a sequence of data.

Project Overview

For this project, I wanted to test the efficacy of LSTM networks in predicting electricity demand in Texas. Specifically, I compared the prediction error of a LSTM network against an FFN, which I consider a baseline neural network. Networks with a *sigmoid* transfer function in the hidden layer followed by a linear output function are good function approximators. For the dataset, I utilized hourly load (demand) data for 2017 from the Electric Reliability Council of Texas (ERCOT), the wholesale market and grid operator for the state. Texas is unique among U.S. states in that its grid is largely isolated from the rest of the country, making it a good test case. In addition, I downloaded historical weather data for the same time period since it has an impact on electricity demand.

ERCOT divides its territory into eight geographic regions (Figure 1) [6]. I selected the Coast region as my starting point to test and tune both networks. After finding the optimal set of parameters, I trained and tested the networks on each other region to provide a robust conclusion.

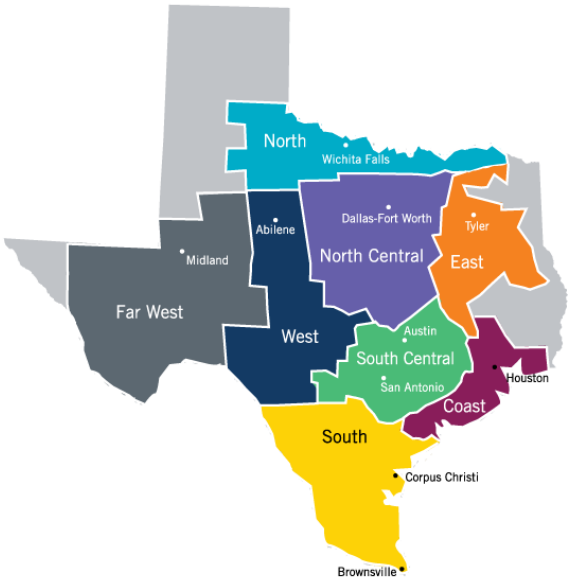

Figure 1

Process

I used the problem-solving framework taught in the Intro to Data Science course to structure my research. This involves a five-step process: crafting a research question, conducting exploratory data analysis, building a model to answer the question, interpreting the output of the model, and communicating those results.

Research Question

The first step in this process is formulating a research question that is **S**pecific, **M**easurable, **A**nswerable, **R**elevant, and **T**ime-specific (SMART).

In this case, the question is specific in that it seeks to identify whether the LSTM network is better than the FFN by direct comparison. The question is answerable because this comparison can be princely measured if we define the better network as the one with lower prediction error. Several error statistics exist. I chose mean absolute error (MAE) and mean absolute percentage error (MAPE). MAE is useful because we report it in the units of the variable of interest, providing an easy refence point.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a_i - t_i|$$

where a_i is the network output and t_i is the target value.

However, because each ERCOT region has a different demand profile, with a wide variance in the ranges of the load measurement (Table 1), I also calculated the MAPE. Since MAPE is a percentage, it allows for easier comparison between regions.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|a_i - t_i|}{t_i}$$

The research question is time-specific in that I chose one year's worth of data to examine and relevant because, as discussed in the opening section, the better we can predict electricity demand, the easier it is to implement cleaner energy solutions.

Exploratory Analysis

After defining the question, but before conducting exploratory analysis, I had to clean the data. This involved converting the date field into the correct format so that Python and Pandas understood them as time measurements. I also had to match the weather data readings with the correct ERCOT region. I used historical weather data from NOAA's local climatological dataset, which reports readings by weather station. To join the appropriate readings with each ERCOT region, I selected the airport weather station from the city indicated in each region on the map shown in Figure 1. These are the cities that ERCOT uses for its regional reference weather data. For the South Central region, I used Austin, and for the South region, I used Corpus Christi.

The next step was to conduct exploratory analysis so that I had a sense for the shape of the data. Electricity load is measured in megawatts (MW), and the ERCOT data are reported at hourly intervals. Each region had different load characteristics. As expected, the Coast and North Central regions, which together host Dallas-Fort Worth and Houston, have the largest loads, followed by the South Central region, which contains Austin and San Antonio (Table 1). Examining load against day and mean load over time-of-day, we see that electricity usage increases during the summer (as a result of air conditioner usage) and during the late afternoons (Figure 2, Figure 3).

Table 1

Electricity Load by Region (Megawatts)

	Coast	East	Far West	North Central	North	South Central	South	West
count	8,760	8,760	8,760	8,760	8,755	8,760	8,760	8,640
mean	11,930.91	1,399.52	2,404.96	13,031.03	818.28	6,575.24	3,466.99	1,153.80
std	2,569.64	314.36	241.92	3,386.23	165.62	1,739.08	831.31	214.43
min	7,422.59	806.16	1,957.94	7,620.74	543.41	3,751.32	2,019.13	802.54
max	20,100.76	2,415.88	3,164.19	24,313.21	1,393.51	11,970.20	5,845.28	1,901.94

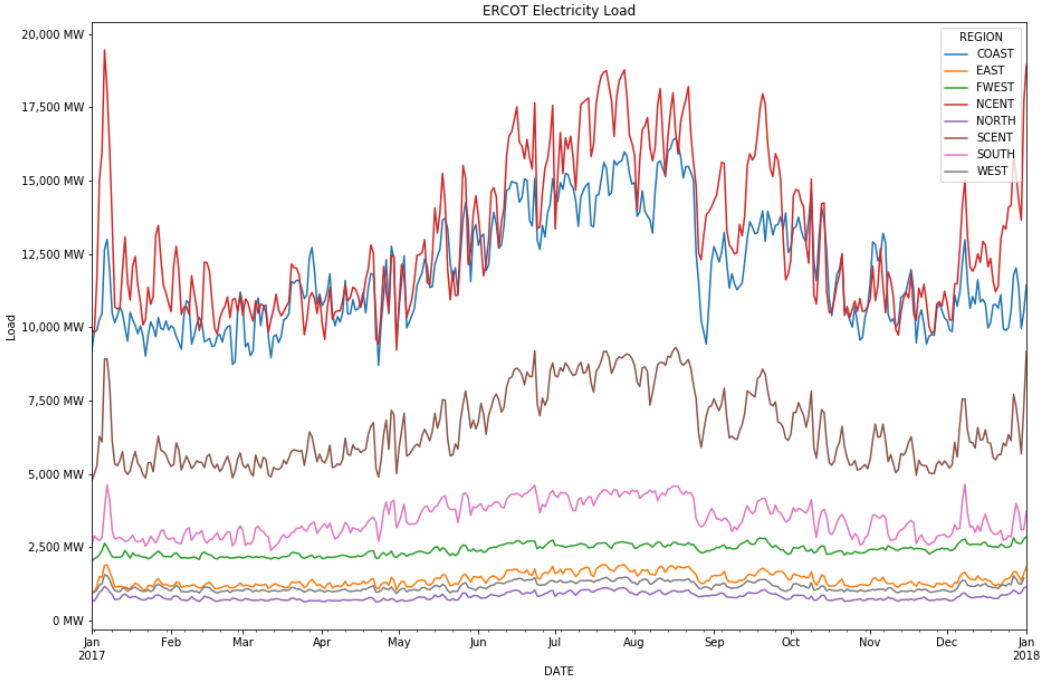

Figure 2

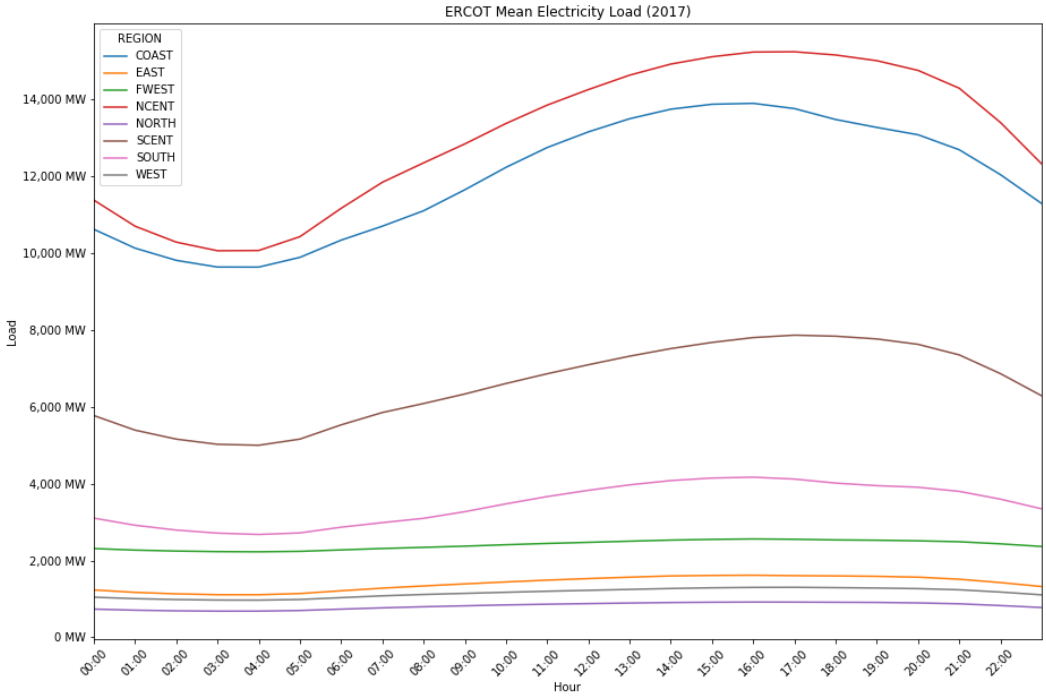

Figure 3

Since I observed this clear daily seasonal pattern, as part of the data preprocessing phase I differenced the dataset, subtracting each hourly reading from the one 24 hours prior. This filters out movements in the data that can be explained by time, leaving the pattern that we are trying to associate with other variables (Figure 4).

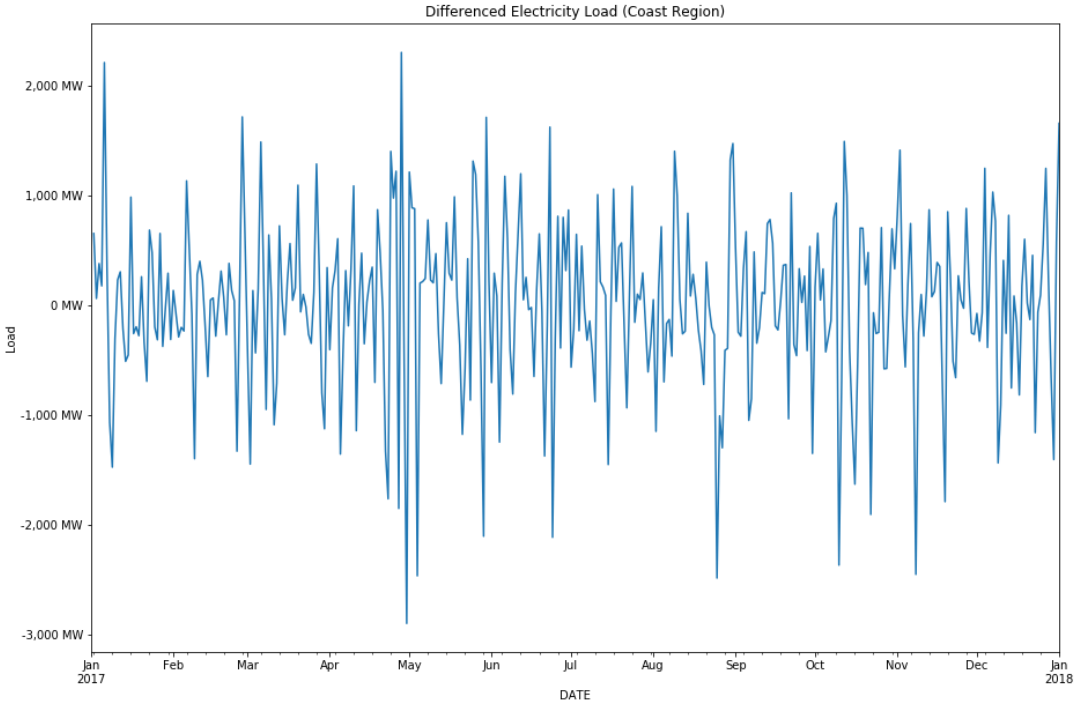

Figure 4

I also normalized the data before running them through the model, which I will discuss in the next section.

I tested the correlation of several weather indicators with the electricity load to determine which indicator to use in the model (Table 2). Based on the results, I selected the dry bulb temperature (the temperature without considering the amount of moisture in the air) to use in the model (Table 3).

Table 2

Correlation

	Dry Bulb Temp (C)	**Humidity**	**Precipitation**	**Wet Bulb Temp (C)**	**Wind Speed**
Load	0.707	-0.328	-0.023	0.579	0.160

Table 3

Dry Bulb Temperature by Region (Celsius)

	Coast	East	Far West	North Central	North	South Central	South	West
count	8752	8744	8747	8756	8745	8751	8744	5826
mean	18.56	16.35	12.61	16.08	13.47	17.29	20.04	14.81
std	6.40	7.42	7.04	7.50	8.09	6.97	6.05	8.03
min	-6.90	-10.40	-12.00	-10.40	-14.40	-8.80	-5.30	-11.40
max	27.30	27.40	24.00	28.00	26.60	26.80	28.60	27.00

The Model

Once I completed the data cleaning and preprocessing, I had to construct a model to solve the problem. Since the goal is to compare the performance of the two types of network, I had to first find the best version of each one, by testing multiple scenarios with different hyperparameters. I started with the FFN (Figure 5).

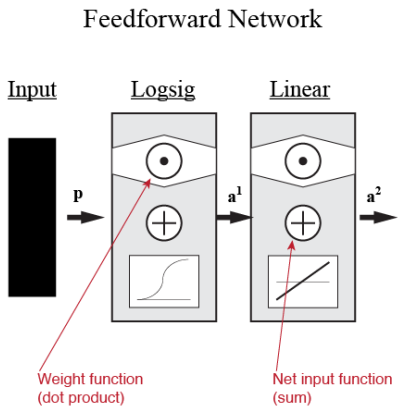

Figure 5

A forward pass through the network is calculated as follows:

$$\boldsymbol{a}^1 = logsig(\boldsymbol{W}^1\boldsymbol{p} + \boldsymbol{b}^1)$$

$$\boldsymbol{a}^2 = \boldsymbol{W}^2\boldsymbol{a}^1 + \boldsymbol{b}^2$$

where $\boldsymbol{W}^m$ and $\boldsymbol{b}^m$ represent the weight matrix and bias vector, respectively for layer m, and $\boldsymbol{p}$ equals the network input vector. Because the FFN in this network performs a time series regression, the input vector contains lag variables so

$$\boldsymbol{p} = [t_h \;\; l_h \;\; t_{h+1} \;\; l_{h+1} \;\; t_{h+2} \;\; l_{h+2} \ldots t_{h+m} \;\; l_{h+m}]^T$$

where t_h and l_h are temperature and load, respectively at hour h for $m = 0, 1, 2$ ... and the network output ($\boldsymbol{a}^2$) is the predicted value for l at hour $h + 1$. I tested several periods of lag variables.

Because the *sigmoid* function outputs a value between 0 and 1, it is good practice to normalize the input data into the same range. Otherwise, it is possible for the gradients to become very large or small during the optimization process, which can slow training. Before training the network, I performed the following normalization operation on the inputs and target values:

$$x_{norm} = \frac{x - x_{min}}{(x_{max} - x_{min})}$$

After developing the FFN model, I turned to the LSTM network. This network can store information about each input that is presented so it builds up a "memory" of the sequence of data. It does this by passing the input with tapped delays of the output through a series of gating functions that determine how much of the prior input to keep along with the new signal (Figure 6). The input gate determines how much of the current input to introduce to the network; the forget (or feedback) gate calculates whether to clear the previous state of the LSTM cell. The results of these two gates are combined in the constant error carousel layer. The output gate governs the output values of the cell [7] [8].

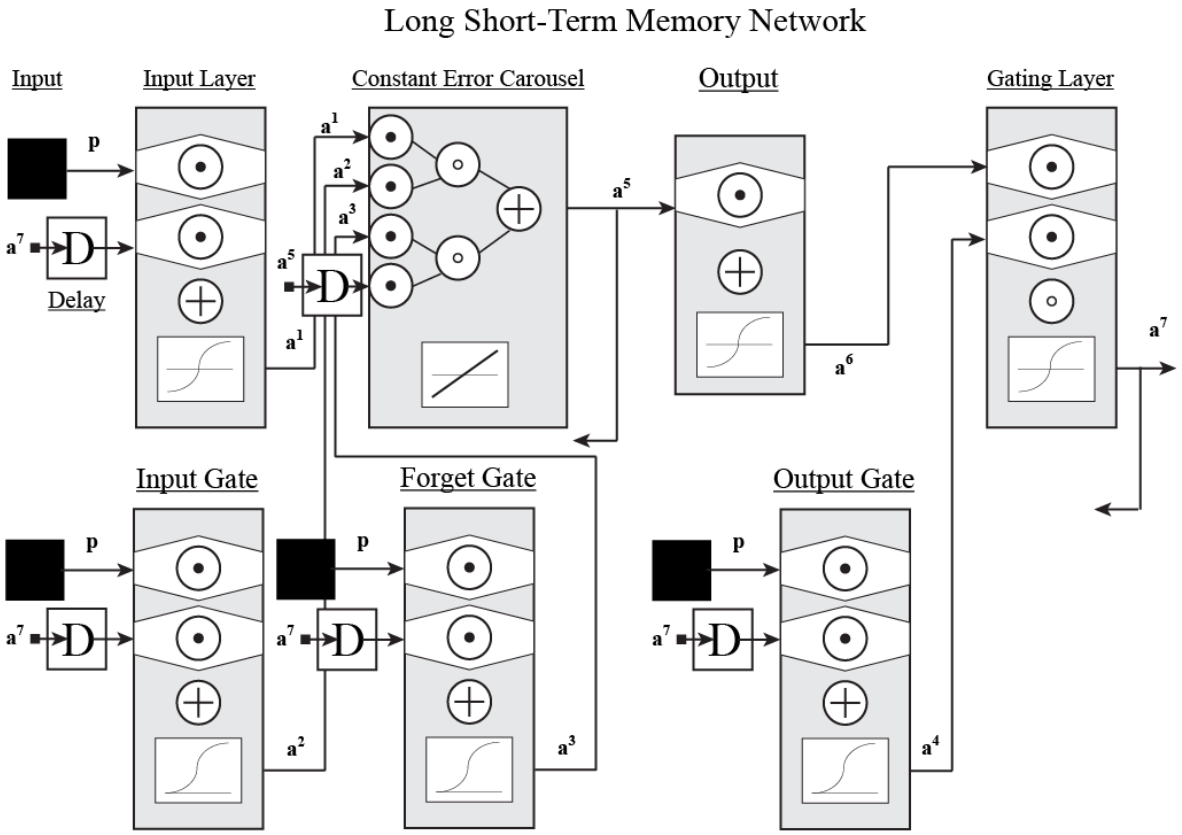

Figure 6

I implemented the network with the built-in LSTM function in Keras running on TensorFlow, which expects each input as a matrix:

$$\boldsymbol{P} = \begin{bmatrix} t_h & \cdots & l_h \\ \vdots & \ddots & \vdots \\ t_{h+m} & \cdots & l_{h+m} \end{bmatrix}$$

Thus, the total input dataset is a three-dimensional array (*samples* × *time steps* × *features*). As with the FFN, I normalized the dataset before training. This time, since the LSTM uses the *tansig* function in its input and output layers, I normalized the data into the range [-1, 1]:

$$x_{norm} = \frac{2(x - x_{min})}{(x_{max} - x_{min})} - 1$$

For both networks, instead of using the standard gradient descent optimization algorithm, I employed the Adam method. Adam works by incorporating the mean and variance of the gradient (moderated by decays) into the parameter update rule [9]:

$$\boldsymbol{W}_k = \boldsymbol{W}_{k+1} - \alpha_k \frac{m_k}{\sqrt{v}_k + \hat{\varepsilon}}$$

where

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1)\nabla f$$

and

$$v_k = \beta_2 v_{k-1} + (1 - \beta_2)(\nabla f \circ \nabla f)$$

where m and v are the mean and variance of the gradient, respectively, while β_1 and β_2 are decay parameters close to 0. $\hat{\varepsilon}$ is a small scalar value to prevent division by 0. Finally, α_k is the learning rate and is calculated by:

$$\alpha_k = \alpha \frac{\sqrt{1 - \beta_2^k}}{1 - \beta_1^k}$$

with α equal to a small positive value. Kingma and Ba recommend $\alpha = 0.001$.

Results

The fourth and fifth steps in the problem-solving framework are to interpret and communicate the model results.

To train both networks, I set aside 15% of the Coast region's dataset for validation and trained on the remaining 85%. I tested different combinations of hyperparameters to find the best version of each network to use for the comparison. Table 4 contains a sample of some of these trials. Additionally, because the networks are initialized with a random set of weights, each training run with the same hyperparameters will yield a slightly different result. Thus, I repeated each run 50

times and reported the median error statistics. Figure 7, as an example, indicates the distribution of errors from the LSTM network's best set of hyperparameters. Once I found the best result for the Coast region, I trained each network on the other regions using the same hyperparameters.

Among the variations of the FFN that I tested, I found that the network with five hidden layer neurons performed best. Adding additional neurons did not increase accuracy and slowed training. The network converged at 125 epochs (with a batch size of 100) so I stopped training at this point. Though this was the best version of the FFN, the MAPE was 5.9%, and when comparing a plot of the actual versus predicted values, we can observe many deviations (Figure 8).

Table 4

Sample of Hyperparameter Combinations (Coast Region)

Network	Epochs	Hidden Layer Neurons	Lags	MAE	MAPE
FFN	100	1	1	648.95 MW	5.93%
FFN	125	1	1	653.38 MW	5.97%
FFN	125	5	1	646.37 MW	5.90%
FFN	150	1	1	655.66 MW	5.97%
FFN	125	5	7	654.82 MW	5.98%
FFN	150	5	7	659.78 MW	6.02%
FFN	150	100	7	6.90.76 MW	6.31%
FFN	250	10	24	671.09 MW	6.13%
FFN	500	10	24	665.58 MW	6.07%
LSTM	5	5	N/A	212.23 MW	1.97%
LSTM	5	1	N/A	250.34 MW	2.34%
LSTM	10	1	N/A	404.09 MW	3.82%
LSTM	15	1	N/A	517.99 MW	4.87%
LSTM	50	1	N/A	686.26 MW	6.26%

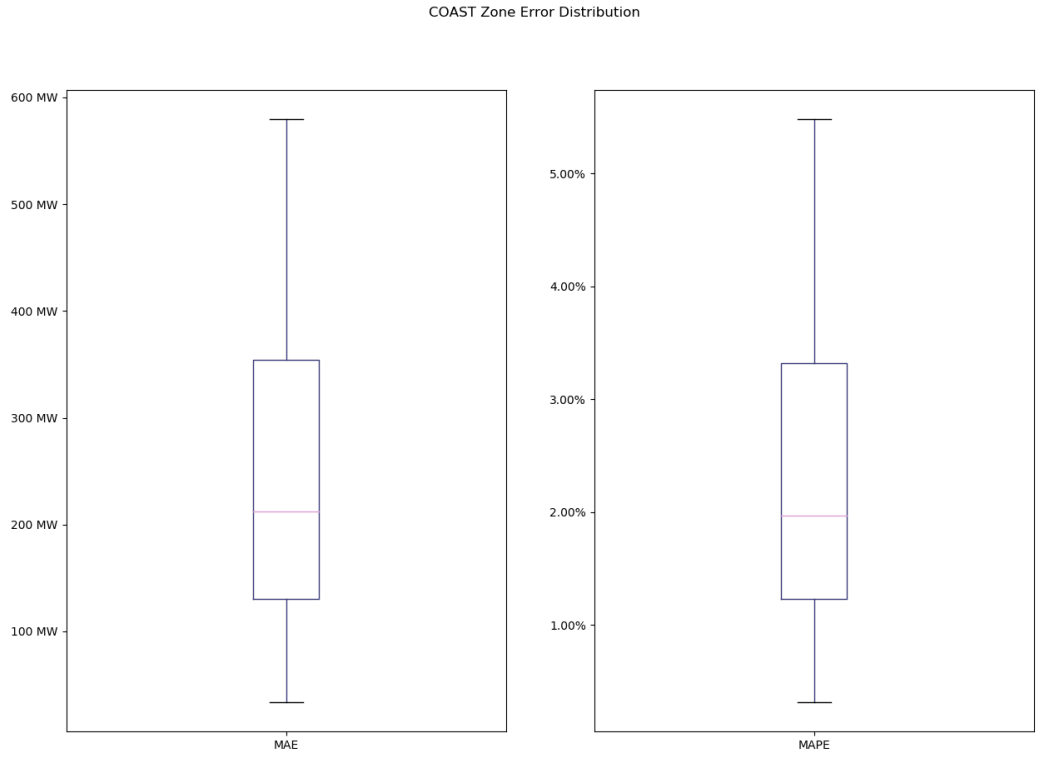

Figure 7

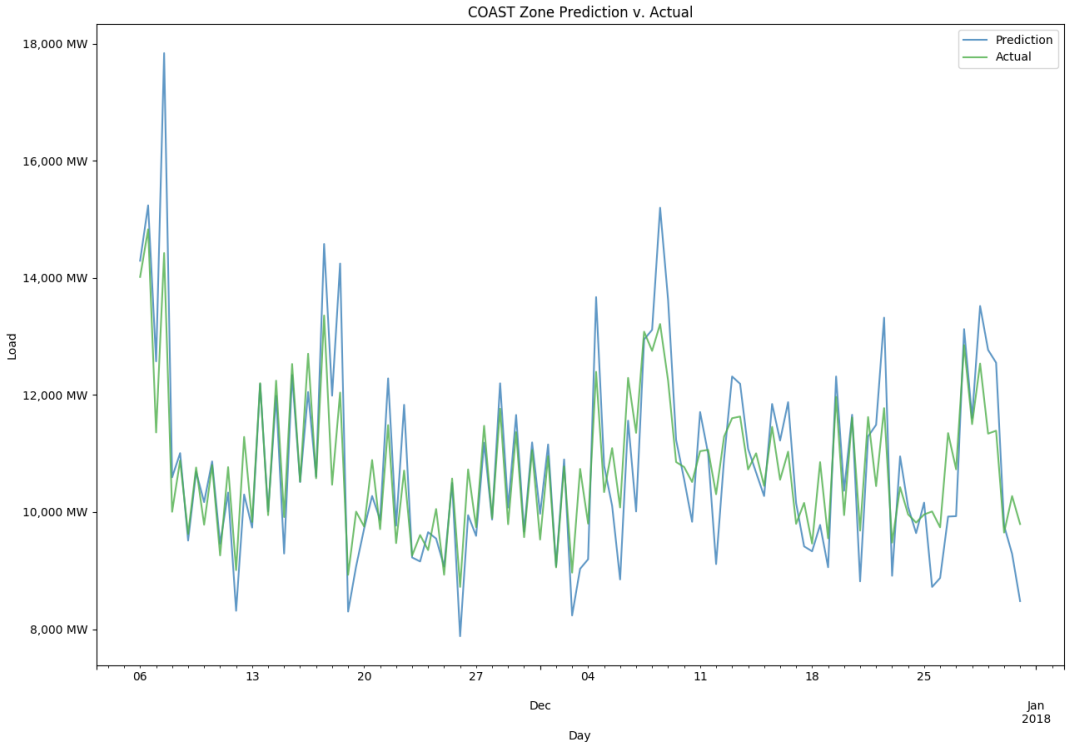

Figure 8

The LSTM network converged at 5 epochs (also with a batch size of 100). Additional epochs caused overfitting that decreased accuracy. I found this fact surprising, and it supports the effectiveness of LSTMs at quickly learning sequential patterns. With a MAPE of 1.97%, the LSTM outperforms the performance of the FFN. We can also observe the increased accuracy by examining the plot of prediction versus actual values (Figure 9).

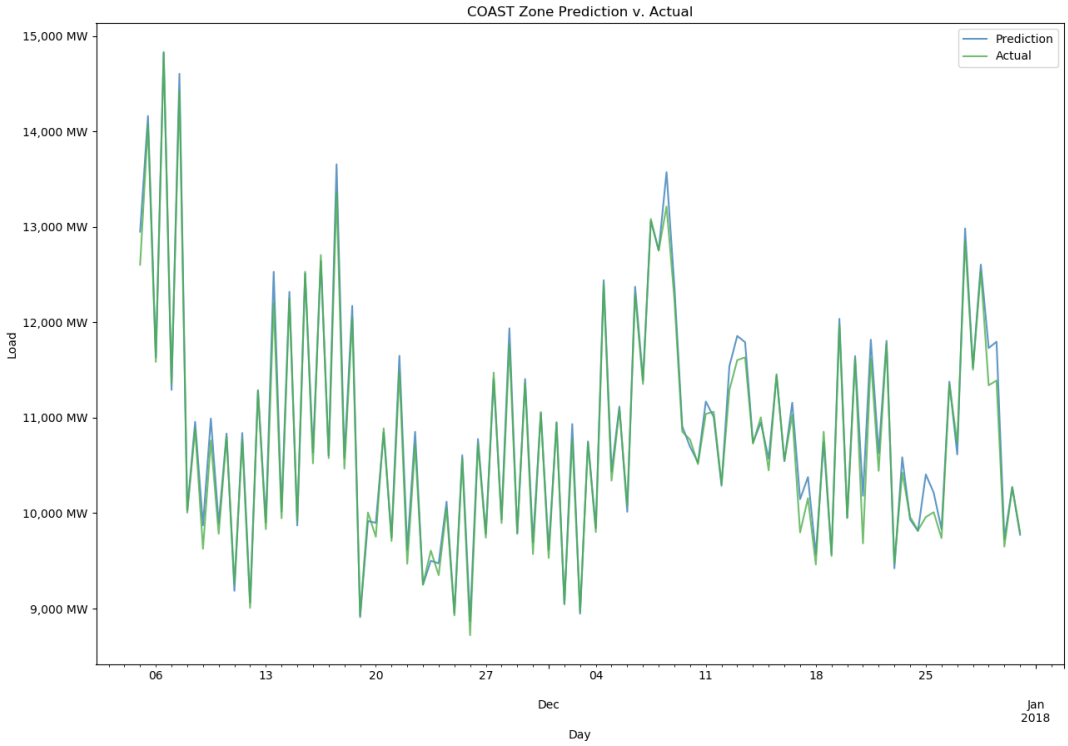

Figure 9

Table 5 provides a full comparison of the FFN against the LSTM for each ERCOT region. In each case, we can see a significant improvement when using the latter network.

Table 5

	Feedforward Network		Long Short-Term Memory Network	
Region	**MAE**	**MAPE**	**MAE**	**MAPE**
Coast	646.37 MW	5.90%	212.23 MW	1.97%
East	94.62 MW	7.09%	25.42 MW	1.94%
Far West	74.06 MW	2.91%	19.30 MW	0.77%
North Central	1,021.40 MW	8.39%	277.38 MW	2.33%
North	49.88 MW	6.19%	13.01 MW	1.69%
South Central	489.93 MW	8.07%	122.82 MW	2.11%
South	250.73 MW	7.70%	66.42 MW	2.14%
West	80.29 MW	6.91%	19.58 MW	1.82%

To extend my work, the LSTM network could be improved by incorporating more time steps (beyond one). In addition, we could test to see if the network is able learn the seasonal patterns within the data, which would eliminate the need for differencing. Beyond the daily patterns, the data likely also exhibit monthly seasonality, which could be learned by training the network on multiple years. Finally, additional input variables, such as economic indicators, could be added to increase the information available for prediction.

The GW Data Science Program

Reflecting on how the George Washington University Data Science program prepared me for this project, a number of points stand out. I had no exposure to machine learning prior to taking Dr. Amir Jafari's two courses. I found both them to be extremely interesting and was excited to challenge myself with a machine learning-based project for my capstone. Beyond machine learning, I used skills I learned in Dr. Nima Zahadat's Data Mining course to perform the exploratory analysis on my dataset. Specifically, I employed Jupyter Notebook running Python to perform the data transformations, preprocessing and to generate summary plots.

I also drew on skills from the Cloud Computing class to set up an AWS EC2 instance to run the neural network training and testing. With the instance running Ubuntu, I installed Anaconda's Miniconda tool to serve as package manager and then set up Python with the pudb debugger tool.

As an extension of my coursework, I taught myself several new topics for this project. First, in the Machine Learning II course, I primarily used PyTorch to implement neural networks in Python. For this capstone project, I decided to try using Keras, running on TensorFlow. Since Keras contains higher level functions than PyTorch, and TensorFlow itself, I could create a network with very few lines of code. This is handy as a time-saving measure, though I found PyTorch to be very elegant and straightforward to debug so I may revert to using it in the future.

Second, while I learned several useful regression techniques in the Applied Economics course, it did not cover time series analysis so I took the opportunity to study that section from the assigned textbook. From this reading, I learned how to work with lag variables, which was necessary for training the FFN. I also learned how to properly handle seasonal trends by differencing the dataset.

Overall, I used this project to apply many of the skills learned over the last two years in the Data Science program while extending them into some new topics. It has been a rewarding endeavor.

Works Cited

[1] U.S. Energy Information Administration, "International Energy Statistics," 2018. [Online]. Available: https://www.eia.gov/beta/international/data/browser/. [Accessed August 2018].

[2] X. Chen, Z. Y. Dong, K. Meng, Y. Xu, K. P. Wong and H. W. Ngan, "Electricity Price Forecasting With Extreme Learning Machine and Bootstrapping," *IEEE Transactions on Power Systems,* vol. 27, no. 4, pp. 2055-2016, 2012.

[3] Y. He, Q. Xu, J. Wan and S. Yang, "Electrical load forecasting based on self-adaptive chaotic neural network using Chebyshev map," *Neural Computing and Applications,* vol. 29, no. 7, pp. 603-612, 2018.

[4] C. J. Ziser, Z. Y. Dong and K. P. Wong, "Incorporating weather uncertainty in demand forecasts for electricity market planning," *International Journal of Systems Science,* vol. 43, no. 7, pp. 1336-1346, 2012.

[5] J. Brownlee, "Time Series Forecasting with the Long Short-Term Memory Network in Python," 7 April 2017. [Online]. Available: https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/. [Accessed July 2018].

[6] ERCOT, "Weather," 2018. [Online]. Available: http://www.ercot.com/about/weather. [Accessed June 2018].

[7] C. Olah, "Understanding LSTM Networks," 27 August 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed July 2018].

[8] A. Jafari, *Long Short Term Memory Network (Lecture Slides),* Washington, DC, 2018.

[9] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, San Diego, 2015.

Datasets

ERCOT**.** 2018. "2017 ERCOT Hourly Load Data." Electric Reliability Council of Texas, Inc. Available: http://www.ercot.com/gridinfo/load/load_hist [Accessed 28 June 2018].

Local Climatological Data (LCD). 2017. Hourly (Jan – Dec 2017) Readings for the following Weather Stations: Tyler Pounds Field TX US, Midland International Airport TX US, Abilene Dyess AFB TX US, Wichita Falls Municipal Airport TX US, Corpus Christi International Airport TX US, Austin Bergstrom International Airport TX US, Houston Intercontinental Airport TX US, DAL FTW WSCMO Airport TX US. (National Oceanic and Atmospheric Administration. Available: https://www.ncdc.noaa.gov/cdo-web/datatools/lcd [Accessed 28 June 2018].