Energy Forecasts via Regression Techniques and Neural Networks

David Troxell
Stanford University
Statistics
`dtroxell@stanford.edu`

Matt Johnson
Stanford University
ICME
`mattj949@stanford.edu`

Abstract

We forecast energy demand 24 hours in the future using various regression techniques and neural network architectures. We find that Gradient Boosting, Random Forests and LSTM neural networks perform well, with the nature of the target demand zone having a large impact on model suitability. Gradient Boosting performed the best by far, yielding the best prediction on a mean-squared-error metric for 17 out of our 20 zones. Surprisingly, Transformer models were not a top performing model in any zone, however we believe there is a lot of opportunity for further exploration with this particular model type.

1 Introduction

Energy demand prediction is a well-researched field, with numerous authors applying a vast number of statistical learning techniques (1). As the nature of the power grid changes, especially considering the widespread adoption of solar panels and electric vehicles, the need to accurately predict energy demand across networks has increased (2)(3). Accurate demand forecasting would allow for incentive structures and readiness preparation across the grid, mitigating the need for costly infrastructure investment and yielding a more stable system.

With this motivation in mind, we task ourselves with predicting energy load across different "zones" in the United States. We are given are the hourly load values for each zone, as well as temperature readings from different temperature stations. Our goal is to accurately predict demand at time points 24 hours into the future for each zone.

2 Dataset

The provided load and temperature data both begin on January 1st, 2004. The load data ends July 7th, 2008 and the temperature data ends June 30th, 2008. There are 20 unique electricity zones corresponding to different geographical regions in the United States. Additionally, there are 11 different temperature values from various temperature stations across the regions; exact locations are unknown. For each hour, we have demand values for all 20 electricity zones and for all 11 temperature stations. The electricity demand values - i.e. the target variable in our problem - have various weeks missing as the dataset, as these missing values were used in a 2012 Kaggle competition as targets to forecast (4). We forward fill values from the previous week to fill these missing weeks of data. Our training set contains all of the data from 1/1/2004 to 12/31/2007, with the beginning 6 months of 2008 acting as the test set.

After forward filling the missing weeks of data in the electricity demand data, we created visualizations to examine the relationship between different variables. For example, Figure 1 displays the daily load patterns across each zone in the dataset. We note that the overall daily pattern of load is relatively

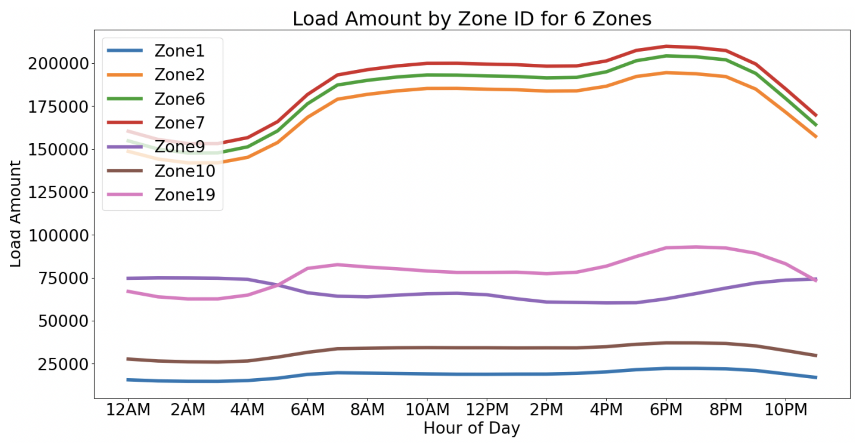


Figure 1: Average hourly load by zone ID for a subset of zones. We see similar patterns for many zones, but some zones exhibit vastly different patterns

similar for many of the zones. However, some of these zones exhibit very different temporal behaviors from the others (see zones 9, 10 and 19). For all zones and for the purpose of our models, we assigned the temperature station whose readings were most highly correlated with the zone's demand. All zones exhibit a fairly large range of demand across the day. Therefore, since zones were self-similar from day to day but dissimilar from each other, we decided to build one model per zone so as to examine each zone's demand pattern independently. We analyze per hour mean, median, and variance for each zone, as well as the trend of temperature readings from each temperature station. While load varies widely by zone and moderately by hour, there is relatively little variation among different days of the week, different months, or different years.

3 Regression Techniques

Our first class of methods were regression techniques. Seven different methods were chosen from this class OLS, Elastic Net, Support Vector Regression (SVR), Random Forest Regression, Adaboost Regression, Gradient Boosting Regression, and K-Nearest Neighbors Regression.

3.1 Data Preprocessing for Regression Techniques

Extensive data reshaping and feature engineering was performed before implementing these regression techniques. Since we train and run each model on each zone separately, we reshaped the dataset so that each observation contained only a single hour of a single day for a single zone. Features included demand and temperature data going back 24 hours to 30 hours, and also 48, 72, and 96 hours as these values had strong correlations with current demand as was observed from the autocorrelation plot that we generated. Other features for the regression techniques included day of the week and month. These variables were one-hot encoded. Additionally, after attempting some of these methods and qualitatively noticing that most models fail to accurately predict hours at "peak" or "valley" demand times of the day (around 5pm-8pm and 1am-5am, respectively) an indicator variable denoting whether the observation lied in these ranges was added.

The last preprocessing step performed was a type of exponential weighting for more recent observations. Since we observed that demand does not vary too widely by year, we construct our exponential weighting parameter to be very small, and time points going back multiple months still have considerable weight attached to them. This scheme results in recent observations having weight of between 1 and 1.5, observations near the middle of the time frame having weight around 0.3, and the beginning of the dataset typically having weight of less than .05.

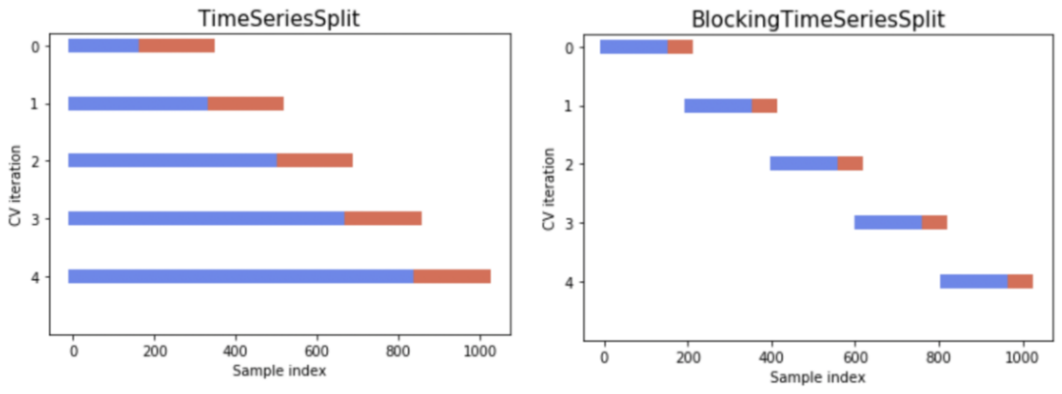


Figure 2: Two different CV schemes tested for regression techniques. Visualization from (5)

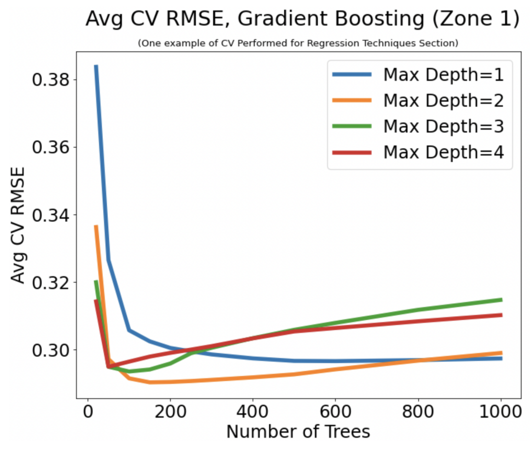


Figure 3: One example of a cross-validation curve produced from our analysis. This plot relates to gradient boosting

3.2 Cross Validation for Regression Techniques

Since these methods contain hyperparameters that can be chosen in model implementation, we decided to execute a systematic cross-validation (CV) scheme. However, we note that the temporal aspect of dataset must be preserved. In order to mimic how new data arrives and how the model will be applied in a real world application, our cross validation scheme cannot use future values to predict past values. We decided to implement two different time-series-based CV processes as we were curious whether final results would be affected by which scheme was chosen. Both CV schemes are depicted visually in Figure 2.

While the CV process depicted in the left plot in Figure 2 was readily available in python, we had to custom-implement the scheme shown in the right plot. While the results are not explicitly shown in this report, we found that final model selection, hyerparameter values, and resulting test error were extremely similar across both CV schemes. For the final results presented in this report, we used the "BlockingTimeSeriesSplit" process depicted on the right due to quicker training times. The hyperparameters that were tuned for each model are displayed in Table 1, and one example of a cross validation curve that was produced in our analysis is presented in Figure 3.

We see in Figure 3 that when holding the interaction depth fixed, as we increase the complexity (i.e. number of trees) of the model, the model initially generalizes to new data well but then beings to overfit as expected. While not displayed, similar plots were produced for the other regression techniques.

Model	Hyperparameters
OLS	Intercept vs. no intercept
Elastic Net	Weights on ℓ_2 and ℓ_1 terms
Support Vector Regression	Kernel type and regularization parameter
KNN Regression	Number of neighbors
Random Forest Regression	Number of predictors and number of trees
Adaboost Regression	Number of trees and learning rate
Gradient Boosting Regression	Number of trees and maximum interaction depth

Table 1: List of employed regression techniques with chosen hyperparameters to tune

3.3 Regression Results and Discussion

For a given zone, after performing cross validation for a particular model and then fitting the model across the entire training set, we froze the model's parameters. This enabled us to simulate using this model in a real-world scenario across our entire testing set. We note that we are predicting 24 hours ahead but the testing set is over weeks of data. Therefore, the true labels for testing data will eventually become input data for future observations (as features include lagged demand values) but the model is not learning any patterns at all from testing data since parameters were solely chosen using the training set. This scheme simply allows us to evaluate how the model would perform had we built it and then decided to employ it for future months in a real-world setting.

Figure 4 displays the gradient boosting model's rolling predictions over one week of the testing set for zone 1. We see that the method is able to reasonably capture both the trend and the seasonality present in the dataset. Feature importance attributes of the model show that the most valued predictor was the demand exactly 24 hours prior to the demand being predicted. While details of the numerical results are provided and discussed in a later section, we note that the regression techniques overall have the power necessary to predict demand within reasonable accuracy. However, we wondered whether explicit time series models and/or more nonlinear models could better capture signal in the dataset. We therefore decided to employ a SARIMAX model and some neural network methods to investigate this further.

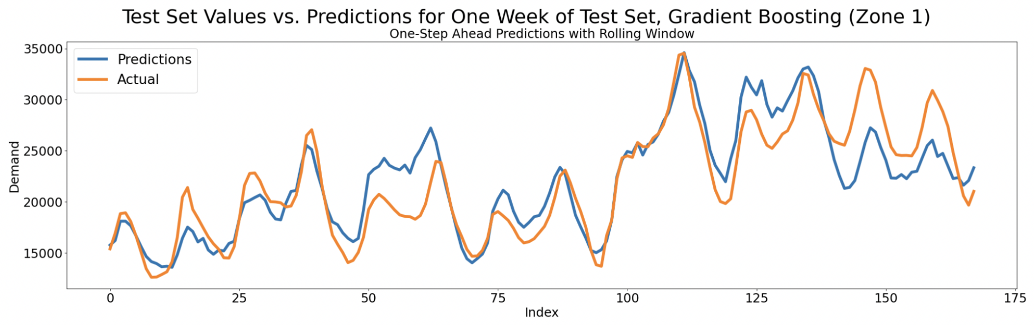


Figure 4: Actual Demand and 24-hour ahead predictions for gradient boosting for one week of the testing set

4 SARIMAX

While the regression techniques mentioned above are quite capable of modeling time series data, we wanted to explore a specific time-series model: SARIMAX, which is an extension of the foundational ARMA model, which forms the backbone of time series statistics. The Auto-Regressive (AR) component captures a dependency on prior values of the process, and the Moving-Average (MA) component captures dependence on prior noise values of the process. p denotes the Auto-Regressive

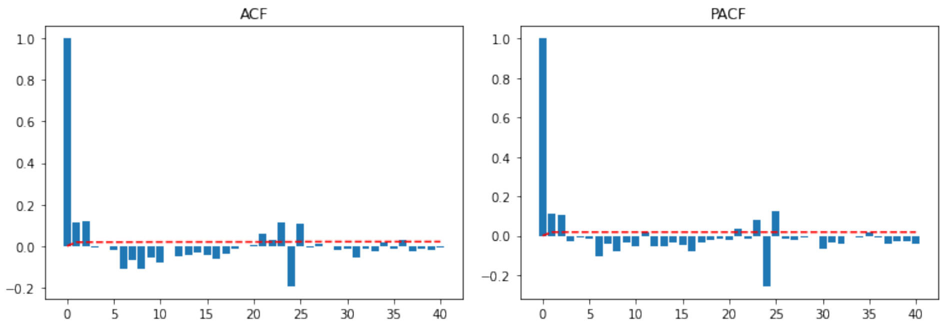


Figure 5: ACF and PACF plots for SARIMAX (2,0,2) $\times$ ($s = 24$) process

order, and p denotes the Moving-Average order. In the equation below, γ and θ denote the AR and MA fitted values respectively.

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \gamma_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-1} \tag{1}$$

SARIMAX models improve on ARMA models in a few crucial ways. First, a seasonal dependence allows for an explicit specification of the time scale on which the process repeats. Integration allows for a differencing of the time series, which may yield a (semi)stationary derivative process of an otherwise non-stationary time series. The X stands for eXogenous variables, which allows for a dependence on external variables; in this project, we include the most-correlated temperature data as our exogenous variable for each electricity load zone. The AR and MA order selection was heavily guided by observing the autocorrelation and partial-autocorrelation plots of the residuals. Our final model was a $p = 2$, $q = 2$ and $s = 24$ SARIMAX model, which indicates an AR order of 2, a MA order of 2, no integration, and a seasonal order of 24. In Figure 5, we see that the residual autocorrelation coefficients are relatively small in magnitude, save for the 24th order terms. However, since we already explicitly include the 24th order AR term in our model, there is little that we could do further to decrease the residual autocorrelation in the data. The red dotted line indicates statistical significance at the 5% level, and we are pleased that the residual correlation magnitudes are small, even if statistically significant.

5 Neural Networks

5.1 Long Short-Term Memory (LSTM)

For our neural network models, we decided to feed in the last 7 days of hourly data to forecast energy demand in 24 hours time. Data processing went as follows: we first split up the date ranges into training and testing windows. Then within each window, we wrote a rolling-window function which created a list of tuples of ((last week of hourly data), (demand in 24 hours)) on which we run the models. Each "observation" is a single tuple from the list.

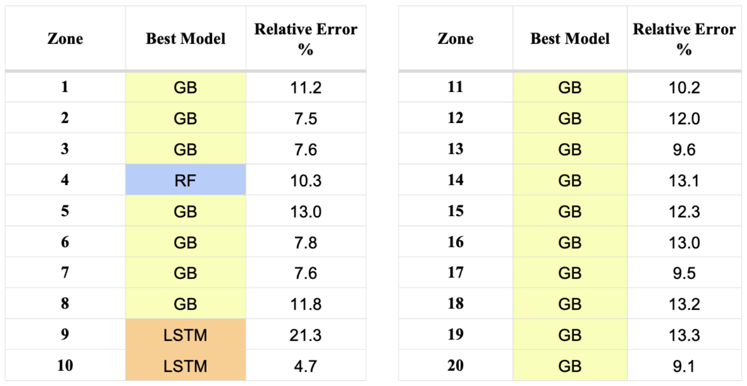
We trained an LSTM as a sequence-to-one model, reading in 7×24 hourly data points and then outputting an estimate for demand in 24 hours. For brevity we do not go into the details of the LSTM model architecture; please see (6) for model details. The LSTM model was 2 layers deep with a hidden-dimension of size 2; we trained for 600 epochs with the Adam optimizer and with weight decay of 10^{-5}, which serves as a l2 regularizer for the model weights. These hyperparameters were chosen by a randomized search over the hyperparameter space for load zone 1. Once chosen, we used the same hyperparameters for all LSTM models. LSTM neural networks have shown a lot of promise in a wide variety of tasks, and the implementation of an LSTM neural network in frameworks such as PyTorch or TensorFlow is very easy. Despite the LSTM model's relatively good performance, we decided to challenge ourselves and explore a more exotic model type: a Transformer.

5.2 Transformers

Transformers were introduced to the world in a now famous paper by Aswani et. al in 2017 (7). This model has shown particular success in sequence-to-sequence applications such as natural language processing. Inspired by this success, we decided to formulate a sequence-to-sequence version of our demand problem on which to apply a modified transformer architecture. Our new problem formulation uses the last week of hourly data to forecast the entire next 24 hours of demand. We modify the Transformer architecture by removing the last softmax later and setting the last linear layer to have output dimension of size 1. The input temporal dimension is $7 \times 24 = 168$ hours, and the embedding dimension is of size 1. Hence, the input is of shape $(168, 1)$, and the output is also of shape $(168, 1)$ for each observation, with an observation being a tuple drawn from our processed dataset. We first apply positional encoding to the 168 unit long input sequence, since otherwise the attention mechanisms on which transformers are built have no notion of the order of the data. We train the model using mean-squared-error, using the last 24 units of the model output to compare against the target 24 hours; we ignore the first 144 output values from the transformer. We use the Adam optimizer and a step scheduler which decreases the learning rate by 5% each epoch. We use early stopping, and generally stop training after around 20 epochs, as the model starts to overfit after this point (validation loss starts to creep up, or at least is non-decreasing).

6 Numerical Results and Model Comparisons

As discussed, we applied a number of models to this dataset. We implemented both linear and nonlinear methods that try to capture the dataset signal in different ways. Figure 6 details the best model for each zone. Additionally, since each zone has demand on different orders of magnitude, we report the relative error percentage per zone in Figure 7, rather than report the RMSE values.



Zone	Best Model	Relative Error %
1	GB	11.2
2	GB	7.5
3	GB	7.6
4	RF	10.3
5	GB	13.0
6	GB	7.8
7	GB	7.6
8	GB	11.8
9	LSTM	21.3
10	LSTM	4.7
11	GB	10.2
12	GB	12.0
13	GB	9.6
14	GB	13.1
15	GB	12.3
16	GB	13.0
17	GB	9.5
18	GB	13.2
19	GB	13.3
20	GB	9.1

Figure 6: Best model for each zone. GB represents gradient boosting, while RF denotes random forest and LSTM is the long short-term memory network

We see that gradient boosting performs best in 17 out of 20 zones. We also note that the cases where gradient boosting did not perform best were the few zones that exhibited vastly different demand patterns from the rest of the dataset. All 3 zones where gradient boosting was not the best performer had relatively flat demand on average throughout the day, and zone 9 even had its peak time around 12am to 4am rather than in the late-afternoon like other zones. The notion that these zones were particularly well suited to different methods (notably the LSTM architecture) reinforces our decision to build different models for different zones. It is evident that we can vastly decrease testing accuracy by not attempting to make a single model that generalizes across all demand zones.

While Figure 6 gives an idea regarding the percent error for the best model for each zone, we can also compare various models to each other. Figure 7 describes, on average, how much worse a given method was from the best method for each zone. While Gradient Boosting has the lowest value, it is not zero because it was not the best performer in all zones.

We see, for instance, that gradient boosting was around 10% worse on average than the best model per zone. This number is due to the fact that in each of the 3 zones where gradient boosting was

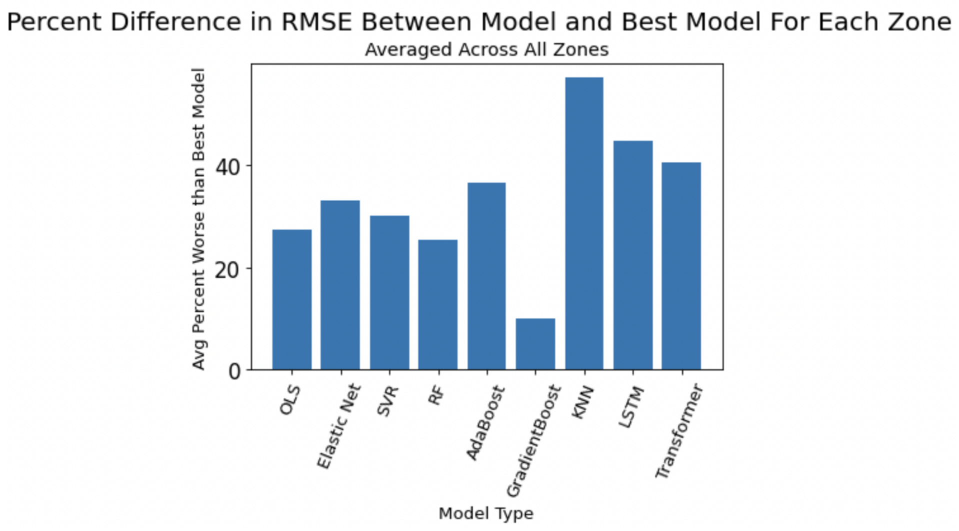


Figure 7: Percent difference in RMSE between each of the implemented methods with the best method for a given zone

not the best method, it was surprisingly around $70 - 80\%$ worse than the best model for the zone. Averaging these values across all zones results in the 10% we see in the figure. Regardless, we clearly find that on average, gradient boosting performs extremely well. Other modeling techniques behaved somewhat similar, besides KNN-regression which exhibited poorer performance measures.

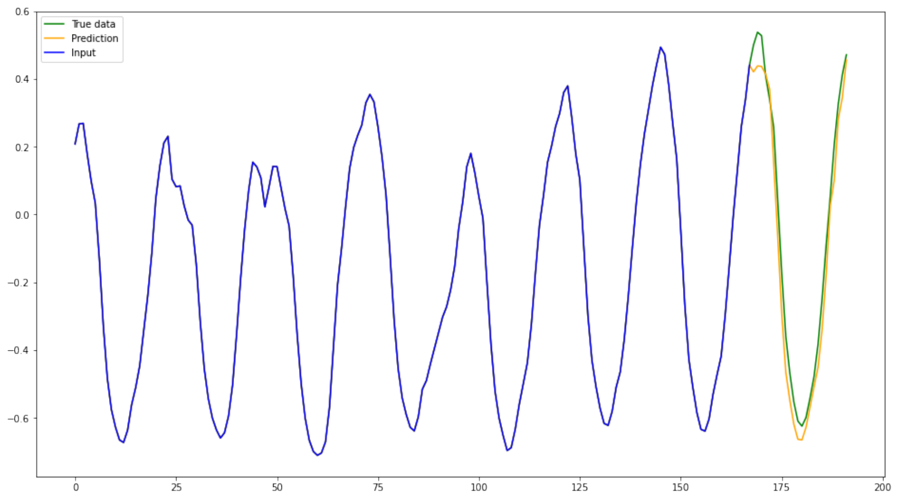


Figure 8: Example Transformer 24-hour forecast from test set; demand zone 1

The 24 hour forecast for the Transformer model is shown in Figure 8. While the qualitative results of the Transformer are quite good, it was not the top performing model on a MSE basis in any of our load zones. The Transformer took an incredibly long time to implement, and as a result we had little time to critically examine and tune the model. In hindsight, there is a major limitation with our implementation. Since we are inputting time-series scalar data, we are forced to pick an embedding dimension of size 1. This severely restricts the power of the model, since the internal parameters are forced to be very small (i.e: have at least one dimension of size 1 rather than something much larger) and this makes the model much less capable of learning patterns in the data. Choosing an embedding dimension of 1 in and of itself severely limited the model, but we also missed an opportunity. Transformers are built on multi-headed attention, and the power of multi-headed attention is to find meaning in different representational subspaces in the embedding and temporal dimensions of the

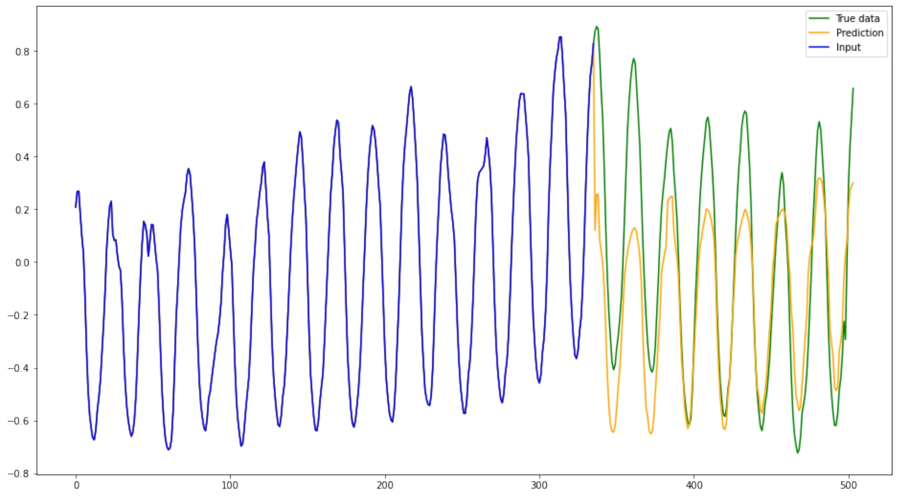


Figure 9: Example Transformer 7-day forecast from test set; demand zone 1

data: this could be different parts of an image if we fed in image data rather than time series data. The idea behind multi-head is for each head (attention mechanism) to focus on separate parts of the input data, and then for later layers to determine which heads have found something useful. In hindsight, this should have been an opportunity to feed in our entire dataset. Each hour could not just contain a single load zone's power demand, but power demand and temperature data cross all load zones and all regions. The transformer might have learned a powerful relationship between all of the zones and regions in this scenario. This would give us an embedding dimension of size $20 + 11 = 31$ rather than 1. Just as PCA can find signal within a noisy high-dimensional input space, a Transformer might have done something similar. Or better still, we might have been able to use an autoencoder to first decrease the dimensionality of the data before feeding it into our modified Transformer.

As an experiment, we also decided to train a Transformer to use the last 2 weeks of data to forecast the following 1 week of demand. As shown in figure 9, the Transformer did a very good job forecasting far into the future. This was absolutely not the case for the LSTM model, which tended towards the mean power demand when used to forecast far into the future. The Transformer seems to have truly learned the seasonality structure of the data, which we found to be very impressive.

7 Conclusion

After extensive data cleaning, data reshaping, and feature engineering, the dataset clearly contained a lot of signal as evident when viewing predicted vs. actual demand in Figures 4, 8, and 9. This being said, the nature of the demand and the relationship present between current and lagged demand values varied widely by demand zone. While gradient boosting performed the best in most settings, sometimes - albeit rarely - more complicated techniques like LSTM exhibited stronger prediction power.

We believe there is a lot of future work to be done as it pertains to Attention and Transformer neural network architectures, as applied to this dataset. We noted our missed opportunities, and it would be fascinating to see how model performance might change with a more careful implementation. Further, predicting 24 hours in the future is not necessary to be useful; shorter time scales are very helpful as well. In this dataset we get regional temperature information, but instantaneous temperature readings of power lines themselves are becoming more readily available to power grid operators. By evaluating the temperature of power lines on a more granular time scale, operators can better plan for peak-demand hours and even immediate contingency events. This necessitates highly accurate forecast for electricity demand over very short term horizons. Using the modeling techniques presented in this report to predict 24 hours ahead is surely valuable to all system operators, but changing these models to predict 15 to 30 minutes ahead with different datasets could prove to be highly beneficial when paired with the use of instantaneous and dynamic temperature readings of power lines.

References

[1] G. K. Tso and K. K. Yau, “Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks,” *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007.

[2] F. Capitanescu, J. M. Ramos, P. Panciatici, D. Kirschen, A. M. Marcolini, L. Platbrood, and L. Wehenkel, “State-of-the-art, challenges, and future trends in security constrained optimal power flow,” *Electric power systems research*, vol. 81, no. 8, pp. 1731–1741, 2011.

[3] B. H. Chowdhury and S. Rahman, “A review of recent advances in economic dispatch,” *IEEE transactions on power systems*, vol. 5, no. 4, pp. 1248–1259, 1990.

[4] “Kaggle: Global energy forecasting competition 2012 - load forecasting,” https://www.kaggle.com/competitions/global-energy-forecasting-competition-2012-load-forecasting/data, accessed: 2022-05-11.

[5] O. Herman-Saffar, “Time based cross validation,” Jan 2020. [Online]. Available: https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8

[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.