Prediction and Analysis of Critical Nodes on the Texas Electricity Grid

Abstract:

A key competitive advantage for all agents participating in decentralized energy markets is to be able to perform accurate forecasting for energy prices. In particular, day ahead prices are of vital importance as they allow energy providers to engage in efficient production, traders to formulate optimal bidding strategies, etc. In this project we explored a deep neural network based approach in order to forecast day ahead prices. We used geospatial node pricing, oil/gas commodities prices, and weather data to come up with day ahead price predictions, which were ended up improving on the utility company's accuracy. Using our best LSTM model, we managed to improve upon the industry standard by an average $0.80 in prediction error.

Introduction:

In decentralized markets, one of the key concerns of energy producers, ISO's (independent system operators) and energy traders is the forecasting of energy prices. In the context of wholesale energy markets, the energy price can displayed as a real time price or a day ahead price. real time prices are determined, as the name suggest, in real time, and it is quite volatile. It represents the shifts of supply and demand, and shortage or overabundance of electricity. Hence the real-time markets operates somewhat similar to stock markets [1]. Day Ahead Prices, on the hand, are computed based on the physical constraints of the grid and the predicted demand and operational costs by the ISO's (the operators of the energy grid) for the next day. So the day-ahead market act as a short-term forward market [1]. In this project, we will focus on predicting the Day Ahead prices.

Normally, the day-ahead prices computed by the ISO´s are the marginal operational costs of the system and are subject to both uncertainties on the physical grid, such as a power lines failing or sudden shortage of demand, and well uncertainties due to virtual trading [2]. As such, forecasting is of extreme importance [1][2]. It allows producers to optimally respond to demand needs and engage in profitable bilateral energy contracts with consumers [3]. Day ahead price forecasts also allow retailers and energy traders to come up with more robust bidding strategies and to stay competitive in the market [3].

Day-Ahead prices forecasting has become an important area of research, since the the deregulation of energy markets. Unlike electricity demand series, electricity price series can exhibit variable means, major volatility and significant spikes [4]. Classical approaches, such as simulation based cost models, regression models and game-theoretical models have been proposed in recent years (see [2][5] for an overview). But none of those approaches seem to work best in every situation of the market and/or the energy grid. Another difficulty faced by those methods is to accurately predict spikes in prices [4]. In these models, usually, price spikes

were truncated before application of the forecasting model to reduce the influence of such observations on the estimation of the model parameters. However, being able to predict spikes turns out to be a key competitive factor.

In an attempt to overcome this challenge, Generalized Autoregressive Conditional Heteroskedasticity (GARCH) process has been tested to simulate price spikes in original price series [6]. However, it appears it was not able incorporate spikes with a height usually observed in the original prices. Spikes have been incorporated into a Markov-switching modes, diffusion models and other data mining techniques [7] have been proposed with varying degree of success. In addition to price spikes, the inherent nonlinear and non-stationary of energy prices still prove to be a major challenge to those classical methods.

More recently, due to a large empirical success of Neural Networks embedded in deep architectures in the fields of pattern recognition, high frequency trading and other areas, energy forecasting models based on Neural Networks have been proposed [8][9][10]. However as shown in the series of works, Neural Network still face some difficulties in predicting such a nonstationary process, such as energy prices. Problems ranging from insufficient input-output data points, too many tuning parameters and the problem of not being able to provide a global model for the whole data history, have been appointed as one of the biggest challenges.

Our project's goal is to contribute to neural network based forecasting in two fronts: The first is from a modelling perspective, as we investigate whether adding oil prices, weather information, and neighbouring bus prices to the input data affects the performance of the network in predicting the day ahead price for a particular node in the energy grid. The second aspect is we try to overcome the challenges present in literature by trying to predict, the day ahead price variations, leveraging the predicted prices that are computed via marginal operational costs computations. Our goal is to use such predicted values to help capture the variation of the energy prices and try to bridge the gap between the cost based predictions and the realized values.

This project report is divided as follows: We first provide a brief description of the energy grid we are considering and then some characteristics of the data we gathered. Then we follow up with presenting our architecture and it’s tuning and training processes. Lastly we provide our experimental results and a discussion on the strengths and weakness of our approach.

Related Work:

Recurrent Neural Network Architectures have seen a lot of recent success in a tackling a variety of time series problems. From financial predictions to working with natural language, recurrent neural networks currently find themselves at the state of the art in performance level for various time-series machine learning problem spaces.

In particular, one popular Recurrent Architecture that has been the center of a lot of attention is the Long Short-Term Memory (LSTM) network. First proposed by Hochreiter et al. in 1997 [11], this architecture surmounts the vanishing gradient problems that is a big issue in training RNNs. Through use of a hidden state and a learned gating mechanism in each LSTM unit, LSTM networks allow information (i.e. gradients) to flow backwards over several timesteps without vanishing, making LSTMs much more capable of learn long range time dependencies.

Work by Jamie Collins et al. compared the effectiveness of various recurrent neural network architectures in regards to their capacity and trainability [12]. They included vanilla RNNs, LSTMs, GRUs, and other variations on these popular architectures in their analysis. Among other things, this research found that LSTMs perform well in regards to both capacity and trainability, and thus are a safe bet when picking a recurrent neural network architecture. Therefore, an LSTM network seemed like a good fit for our problem space.

Description of the Problem:

The electric grid is likely the most complex, poorly understood, difficult-to-predict human creation after the stock market. If one rises from their sleep in the middle of the night and starts their microwave to satiate a guilty craving, the electric grid must perfectly meet that new electric demand. Any less and this hungry individual won't have power, any more and nobody will have power.

The Texas electric grid is made up of approximately 13,000 individual nodes, that serve as waypoints that power travels in between various plants (either solar, wind, coal, etc) and to consumers. Distributed across these nodes are a small handful of S-Points, which are power hubs critical to the grid itself. The utility company in charge of the grid keeps track of real time wholesale prices in 5-10 minute intervals for each node. However, the various power plants also need to know how to ramp production up or down to meet projected energy demand, so the utility company provides predictions of the price of power at each node 24 hours ahead. We attempted to predict the 24 hour prices for two S-Points in the Dallas-Fort Worth metropolitan area using past node price data, weather data, and oil and gas price data.

Sources of the Data:

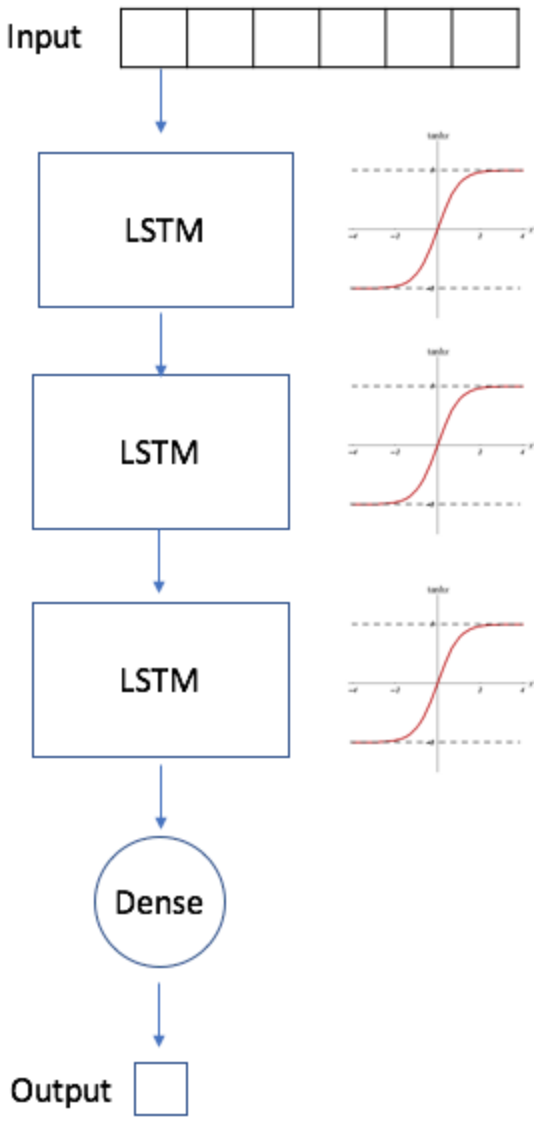
Weather Data:
Weather data was obtained from the Climate Data Online tool maintained by the National Centers for Environmental Information. We specifically used the Global Hourly dataset which included hourly data for temperature, precipitation, dew point, visibility, and several other weather data features. The data we used from this dataset was comprised of continuous values so we used these data directly as input features for the neural net. We also augmented the dataset with features that represented hour of the day, day of the month, and month of the year.

Our hypothesis is that these extra features would make it easier for the neural net to interpret the cyclical nature of this data, which is closely linked to time of day and month of the year.

Node Price data:
The node data was pulled with permission from a database that Liam Purvis created as an independent contractor for Intersect Power. This data itself was cobbled together from a variety of archaic public sources. The day ahead data came via a downloadable link upon request, the S-Point data was in excel format from another requestable link, and the real time prices was actually mailed in a flash drive, stored in triply nested zip files. Due to the changing and dynamic nature of the grid, we only gathered data from the timespan of 01/01/2014-05/31/2017.

The Texas node data was far too massive to work with it in its entirety. 13,000 nodes times 3.5 years times 365 days times 24 hours time 6 timestamps is approximately 2.5 billion observations. To manage this, only looked at a random sample of about 50 nodes within 50 miles of these two S-Points and we rounded and averaged all timestamps to the nearest hour,

Gas and Oil prices:
Gas and oil prices were obtained from the from the U.S. Energy Information Administration (EIA). For gas prices we utilized daily spot market prices . For oil prices we utilized the spot prices for the WTI-Cushing crude oil also on a daily frequency. In order to complete the values for weekends and special holidays (when such spot market prices are not available) we utilized a simple linear interpolation scheme to fill in the missing values.



Building and Tuning the LSTM:

Architecture:
We used a stacked LSTM architecture for this deep learning problem. We used three 64-node LSTM layers with tanh activations. The final output layer was a dense layer with a linear output function. The forget bias for each LSTM layer was 1. Multiple different loss functions were tried with this architecture; further information about the impact of loss function on the neural net's performance will be discussed later in this report. We separated the data into 6-hour long segments and trained the LSTM on these 6-hour segments.

Additionally, after this architecture was optimized, we experimented across all possible losses. The table below shows the loss function used in the first column, the mean error of our model minus the mean error of the utility's predictions in the second column, and the standard deviation of our errors minus the standard deviation of the utilities errors in the third column. A way to view table is that the lower the value in the columns, the better our model performs, and negative values imply it is beating the utilities model.

Loss	model_minus_utility_avg	model_minus_utility_std
mean_squared_error	0.251588	0.249808
mean_absolute_error	-0.704015	0.524433
mean_absolute_percentage_error	-0.183956	0.822222
mean_squared_logarithmic_error	3.72275	0.90617
squared_hinge	32.7416	22.7897
hinge	108.504	97.9189
categorical_hinge	11.1472	6.30232
logcosh	-0.753727	0.191243
categorical_crossentropy	4.29568	2.07097
kullback_leibler_divergence	13.0001	5.2229
poisson	6.41472	4.73689
cosine_proximity	3.21612	1.47233

(Note that this was optimized on the validation set, not the final test set).

Finding Good Hyperparameters:

To find good hyperparameters for our LSTM model, we performed hyperparameter sweeps for 3 different parameters to see how different values would affect our models performance with regard to the most relevant loss metric, mean absolute error. The hyperparameters we chose to test include the number of layers of stacked LSTMs, the number of neurons in each LSTM unit, and the number of previous timesteps our model would consider before making a prediction.

In each sweep, we help all other hyperparameters at default values while testing model performance at various different values for the hyperparameter in question. The default values we chose were 3 layers of LSTMs, 20 neurons per unit, and 12 timesteps to consider. While not an exhaustive gridsearch, these sweeps gave us a good idea on how our model was affected by different choices of parameters. Results of these sweeps are shown below.

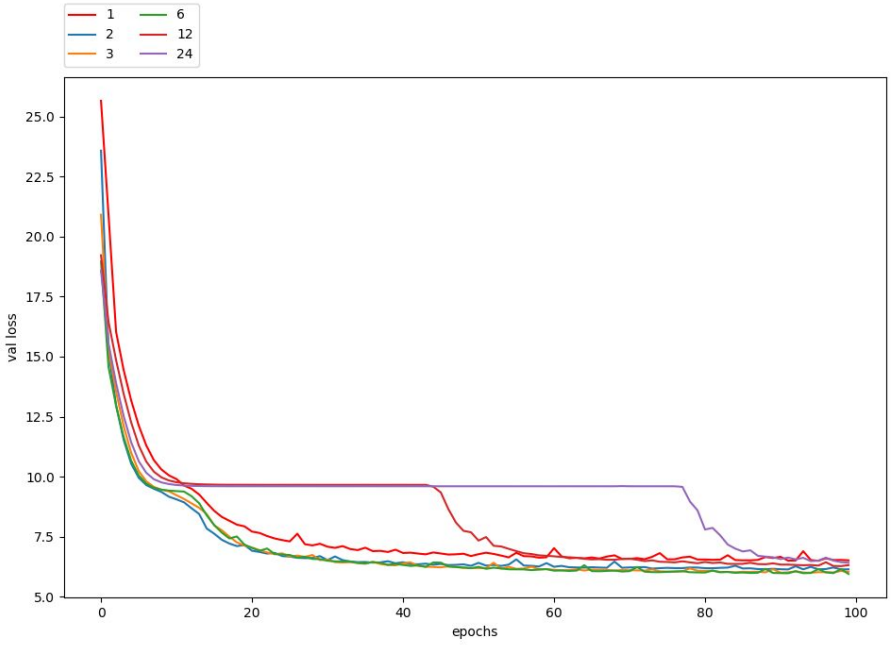


Figure 1: Results of the “Number of Timesteps to Consider” sweep

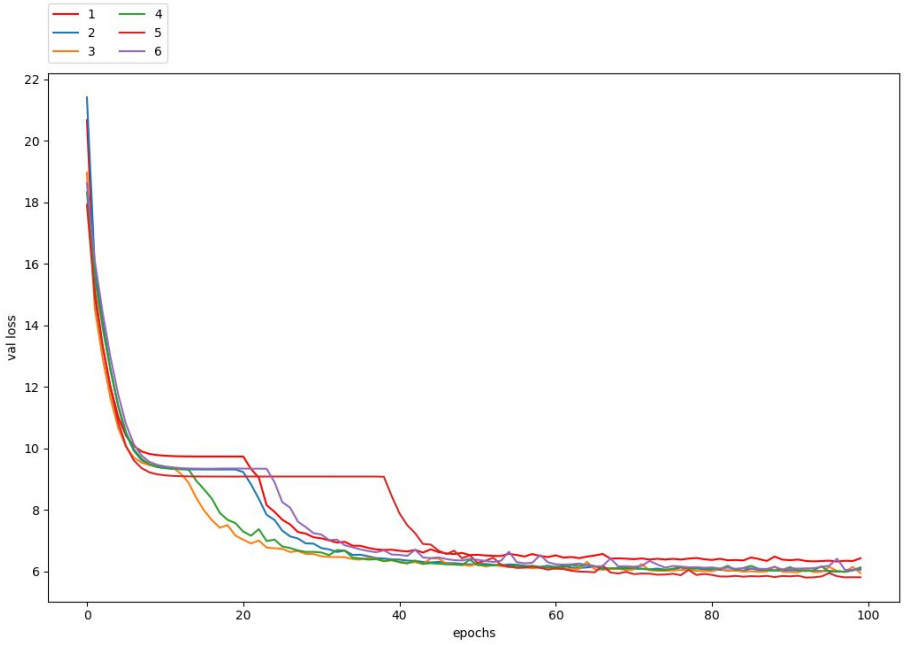


Figure 2: Results of the “Number of Layers” sweep

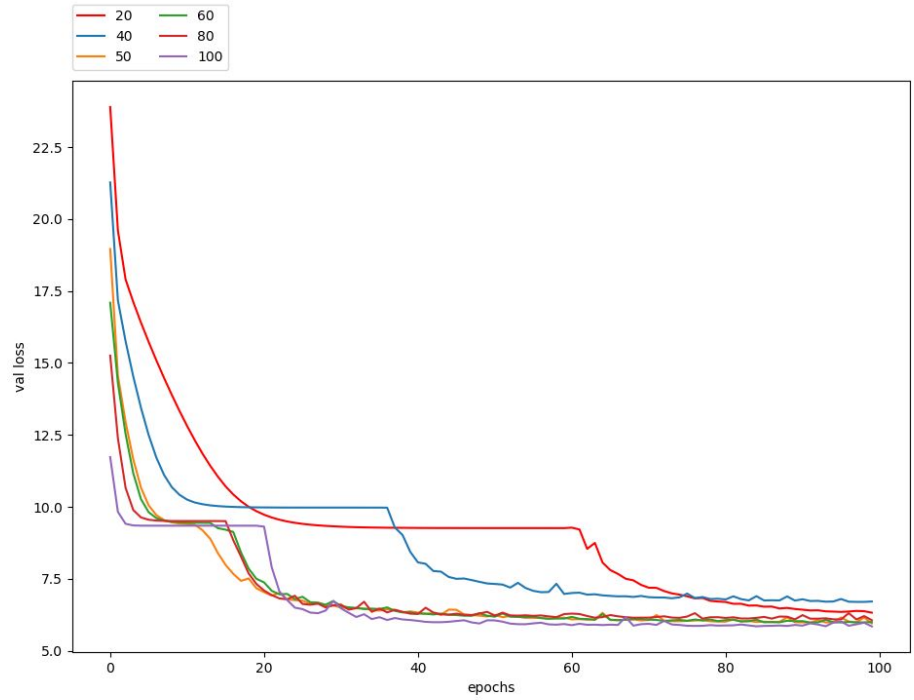


Figure 3: Results from the “LSTM Unit Size” sweep

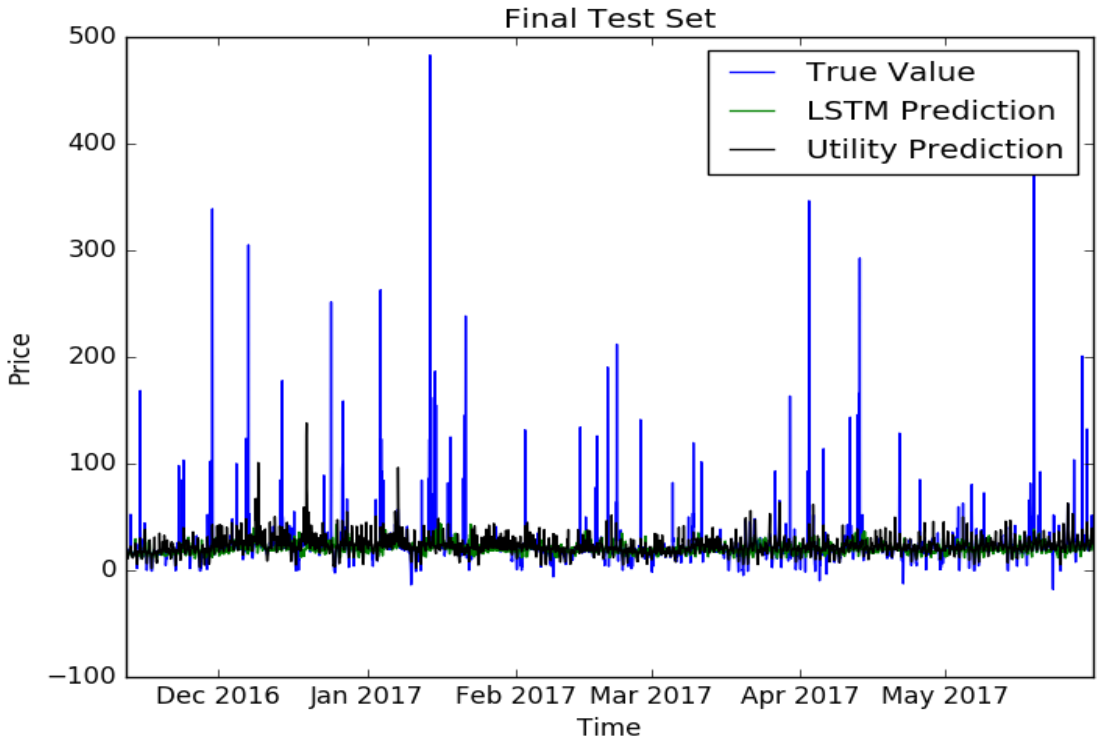
From the timestep sweep, we can see that LSTMs with timestep hyperparameter values of 12 and 24 have a harder time training, whereas a value too low (such as only 1 timestep) may cause a drop in model performance. From the number of layers sweep, we see that a greater number of layers will be slightly slower to train, but may result in slightly improved model performance. From the Unit size sweep, we learn that 20 is probably too small of a value of this hyperparameter, and we achieve better model performance at higher values.

Based on the results of these sweeps, we settled on an architecture with 3 LSTM layers, a unit size of 64, and 6 timesteps to consider. We chose these values to balance good model performance and ease of training.

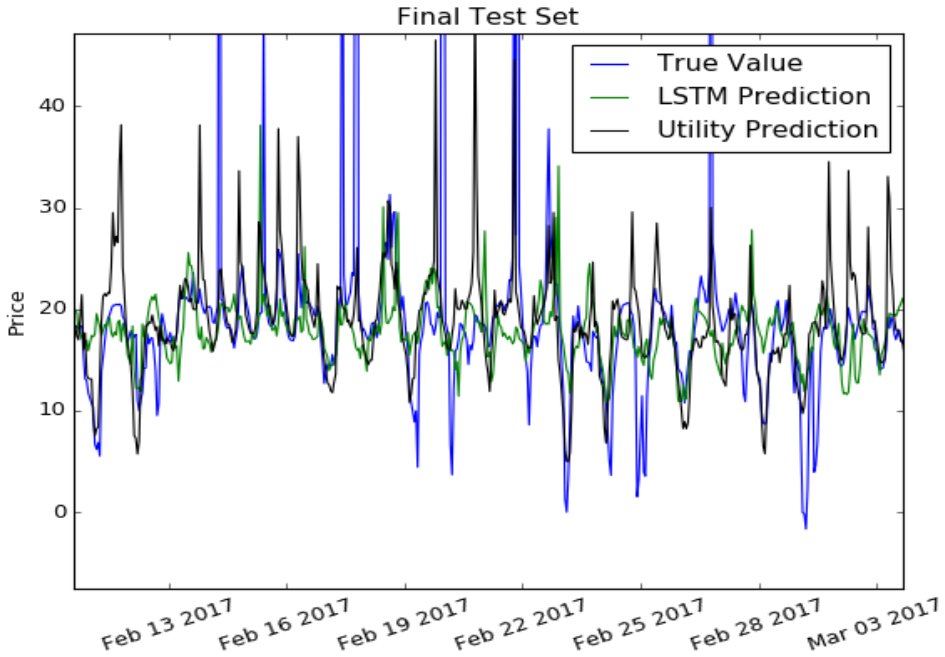
Successes:

After tuning and training on the data spanning from January, 2014 to November 2016, we tested our model on the time period from December 2016 to May 2017. On this test set, our model had an average absolute error of $5.61, while the utility’s day ahead predictions had an average absolute error of $6.41 over the same period. When tuning our model before we were able to beat the utility’s prediction, creating a rudimentary ensemble by taking the mean of the two would result in better performance than each of the predictors individually. However, when we finished tuning, our LSTM model would perform better on its own than the ensemble did. Pur model resulted in an average gain over the utilities predictions of about $0.80 (Note that there was some minor variation in these results as we re-ran our code due to the inherent randomness in neural networks). Considering that these predictions are made every hour, there are about 13,000 modes on the Texas grid, and assuming that this average extrapolates to the

rest of the nodes we would be predicting, we could naively assume that this results in yearly savings of up to $ 9,783,873.
Here is a visual representation of our model on a the entire dataset:



Although it is hard to see much from this distance, it is clear that the extreme outliers are very poorly predicted by both models. Let us examine a section of the time series more closely:



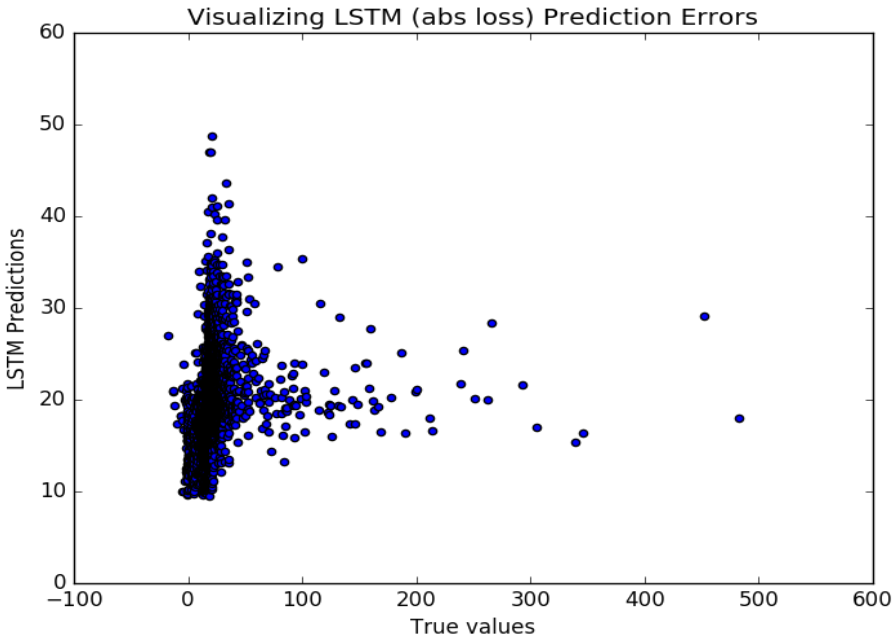
It appears that, even though the outliers are still badly estimated, the utility's model is keen to overestimate the sinal cycle of the true values, while our model seems content to extract most of its accuracy by hugging the true values when they are closer to the mean.

Improvements and Weaknesses:

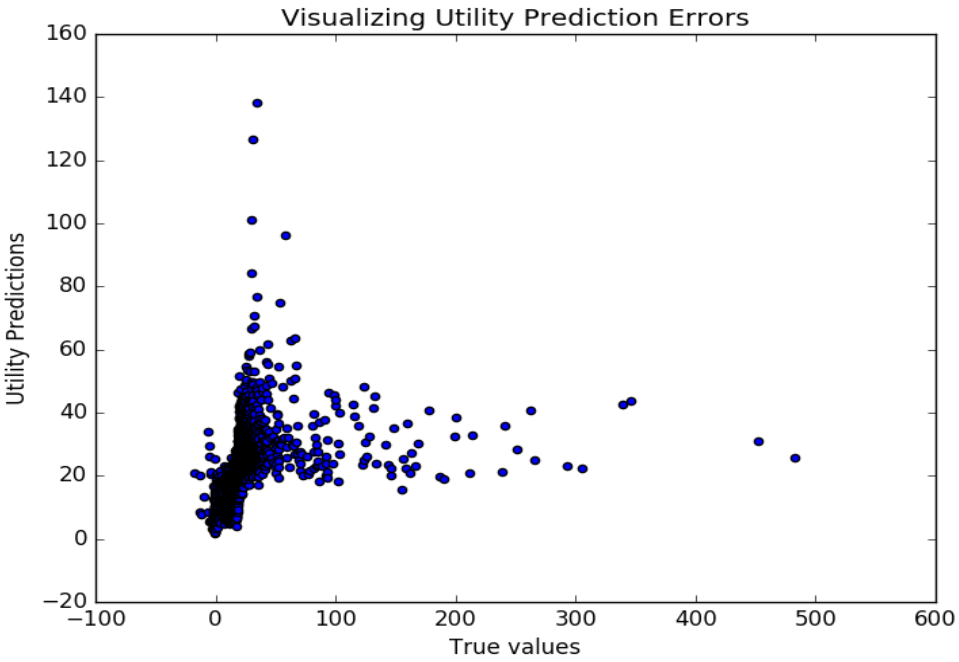
Our model would likely perform much better if it had access to extra data that the utility predictions are partially based off of. For example, we are unsure of which nodes local to the S-Points are most connected and correlated with them, instead relying on a random sample of ~2% of the available nodes within a 10,000 square mile area. Likewise, there is a whole other side to this dataset: the time series of energy production being fed into the grid. While the prices may be viewed as a proxy for demand, the production is the other half of the puzzle. Moreover, much of our weather data and gas price data was highly granular, often priced to the day rather than hour. It is possible that with with these extra data sources the average loss of our model would be reduced significantly.

It should be noted, from the table above in "Building and Tuning the LSTM", that none of the losses used by our model were able to reduce the variance of our prediction errors below that of the utility's. Due to the fact that the grid can easily go down if it is poorly managed, it is quite possible that the utility grid chooses to prioritize reduced variance of its prediction errors over absolute accuracy. Because we weren't able to find the exact loss they were using, we approached this problem with the the goal of extracting monetary value over time rather than managing the volatility between the grid and our estimates.

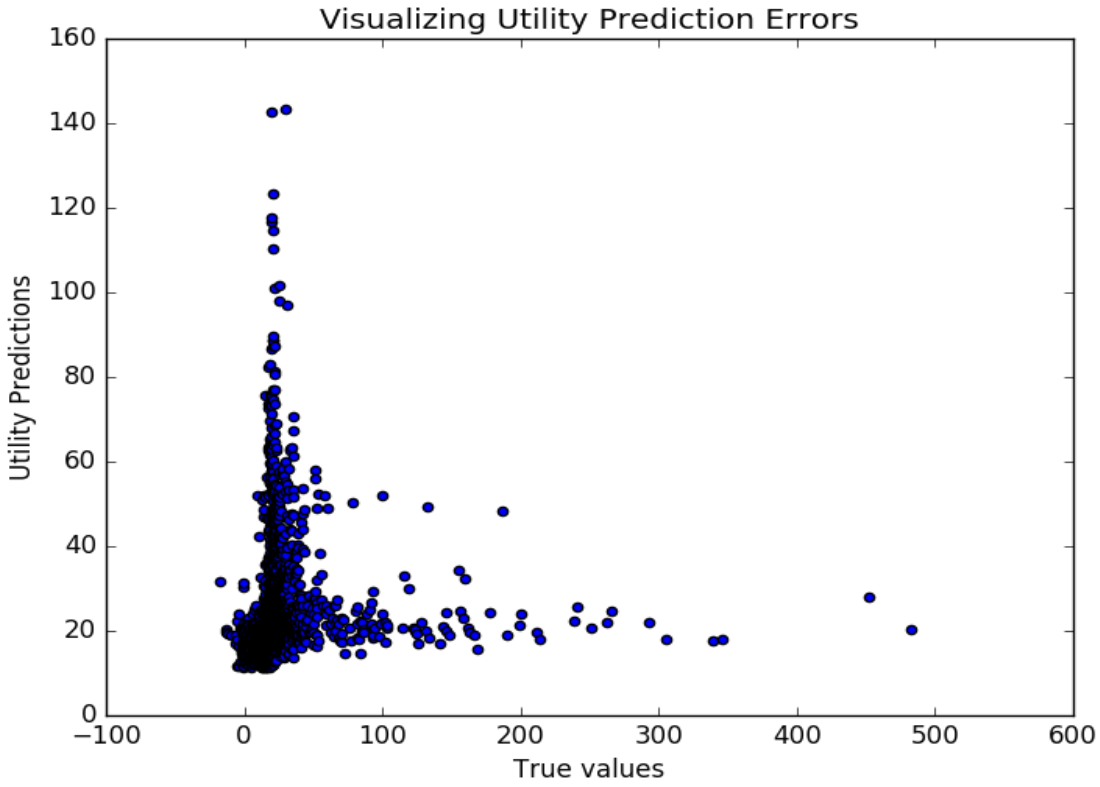
We can, however, guess at the type of loss they are using. It appears to be some kind of squared function. Observe how linear our model's performance is on the test set when we use our logcosh loss:



Similarly, observe the utilities prediction, which introduces a bit of nonlinearity:



Note the similar nonlinearity when we use a mean squared loss:



However, when our model used mean_squared loss it performed noticeably worse than the more linear loss functions (mean absolute and logcosh).

Conclusion:

In this project we trained an LSTM-based architecture in order to forecast the day ahead energy prices for a part of the Texas grid. Despite having insufficient data and virtually no unique insight into this field, we attained indpendent results that came significantly above the industry's model. When comparing our predictions to the utility company's predictions, we attained a significant drop in error that, if implemented, may have the potential to result in tens of millions of dollars of savings per year. Furthermore, we were able to analyze the utility's predictions and compare them against our own, allowing us to guess at aspects of their underlying model.

Bibliography:

[1] Conejo, Antonio J., et al. "Forecasting electricity prices for a day-ahead pool-based electric energy market." International Journal of Forecasting 21.3 (2005): 435-462.

[2] Weron, Rafał. "Electricity price forecasting: A review of the state-of-the-art with a look into the future." International journal of forecasting 30.4 (2014): 1030-1081.

[3] Kwon, Roy H., and Daniel Frances. "Optimization-based bidding in day-ahead electricity auction markets: A review of models for power producers." Handbook of Networks in Power Systems I. Springer Berlin Heidelberg, 2012. 41-59.

[4] Voronin, Sergey, and Jarmo Partanen. "Price forecasting in the day-ahead energy market by an iterative method with separate normal price and price spike frameworks." Energies6.11 (2013): 5897-5920.

[5] Conejo, Antonio J., et al. "Day-ahead electricity price forecasting using the wavelet transform and ARIMA models." IEEE transactions on power systems 20.2 (2005): 1035-1042.

[6]Keles, Dogan, et al. "Comparison of extended mean-reversion and time series models for electricity spot price simulation considering negative prices." Energy Economics 34.4 (2012): 1012-1032.

[7]Zhao, Jun Hua, et al. "A framework for electricity price spike analysis with advanced data mining methods." IEEE Transactions on Power Systems 22.1 (2007): 376-385.

[8] Amjady, Nima. "Day-ahead price forecasting of electricity markets by a new fuzzy neural network." IEEE Transactions on power systems 21.2 (2006): 887-896.

[9] Amjady, Nima, and Farshid Keynia. "Day-ahead price forecasting of electricity markets by a new feature selection algorithm and cascaded neural network technique." Energy Conversion and Management 50.12 (2009): 2976-2982.

[10] Rodriguez, Claudia P., and George J. Anders. "Energy price forecasting in the Ontario competitive power system market." IEEE Transactions on power systems 19.1 (2004): 366-374.

[11] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. Neural Computation , 9(8):1735–1780.

[12] Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. Capacity and trainability in recurrent neural networks. arXiv preprint arXiv:1611.09913, 2016.