CS230

Discovering Decision-Making Patterns of Thermal Electrical Generating Unit Operators

Fletcher H. Passow
Department of Statistics
Stanford University
passow@stanford.edu

Abstract

Fossil-fueled generators still supply most of our electric energy, but economists know very little about the proprietary decision-making processes that drive their control decisions. To help economists better understand these decision-making processes, I used an attention-based deep neural network model to impute gaps in generators' dispatch trajectories. While I did not have sufficient time to extract the explainable AI insights from my trained model, I did modify a state-of-the-art transformer model to make it run 30% faster while maintaining performance.

1 Introduction

Consumers expect reliable electricity, but wind and solar electric power generators produce variable amounts of energy depending on weather conditions. To accomodate more wind and solar generation, electric grid operators must adapt the grid system to intake more energy from variable sources while still outputting reliable electricity. In the near term, operators must utilize electric generation facilities powered by fossil fuels (hereafter "TGUs" for "thermal generating units") to fill in the gaps in renewable energy supply. However, these TGUs were not designed to respond quickly when renewable energy becomes unavailable. Their designers intended for them to operate many hours or days at a time before turning off. To make most effective use of TGUs, the economists who design wholesale electricity markets must understand the decision-making processes that the owners of these TGUs use to turn their plants on and off. Unfortunately, owners of TGUs closely guard their decision-making processes, so economists have difficulty learning about them. The owners of TGUs must place bids in wholesale electricity markets, and secrecy about decision-making can provide a competitive advantage.

To better understand the decision-making processes of TGU owners, I take advantage of a newly accessible public dataset on TGU air pollution and level of power output (called "dispatch" in the industry). I learn TGUs' historical dispatch behavior with an attention-based deep neural network (DNN) model trained in an unsupervised fashion. My DNN model takes as its input a tri-variate time-series of TGU operational measures and outputs imputations of masked portions of that time-series. My model, based on the state-of-the-art Time Series Transformer proposed by Zerveas et al. [2021], extends that work by introducing convolutional layers, enabling it to efficiently digest long time-series. My original intention had been to explain TGU behavior to economists by extracting and visualizing the attention weight patterns produced by my model. However, adapting the model to work with long time-series took longer than expected, so this report focuses on those efforts and defers the explainable AI tasks to future work.

CS230: Deep Learning, Spring 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

2.1 TGU operations

I could not find more than one other paper using the U.S. Environmental Protection Agency Continuous Emissions Monitoring System (CEMS) Data to understand TGU dispatch. This lack of literature is not very surprising, since until recently the CEMS data has been difficult to access. The one paper that I found that does use the CEMS data for a similar purpose did not use machine learning tools at all, but focused on analytics.Park et al. This paper estimated the operational characteristics of a TGU, including its minimum output, its maximum ramp rate (how quickly it can turn on), and its maximum output. It then examined the evolution of these characteristics over time, especially as renewable generation has increased. This paper contributes valuable insights about the physical characteristics of a plant, but it does not capture the decision-making process of the plant operators over time.

2.2 Time-series regression

Vaswani et al. [2017] suggested that self-attention could be used separately from recurrent neural networks. The main benefit is self-attention's ability to capture long-range dependencies. A side benefit is self-attention's interpretability. Attention weights point directly to the parts of the input sequence that the model found to be important in generating the output. However, self-attention does suffer from poor computational scaling on long sequences. For a sequence of length w and a representation dimension d, the self-attention mechanism has an $O(w^2d)$ complexity. Contrast this with recurrent and convolutional approaches, whose complexity scales linearly, not quadratically, with w. Zerveas et al. [2021] adapted the transformer architecture for use with time series. As of their paper's publication in 2021, they achieved state-of-the art performance on several canonical time-series regression and classification problems. They achieved this even with limited training data. I take their approach as a starting point for this project.

Dempster et al. [2020] and Fortuin et al. [2019] suggested alternative time series regression models, ROCKET and SOM-VAE respectively, which competed strongly with the time series transformer of Zerveas et al. Zerveas et al. [2021] ROCKET achieves state-of-the art performance on univariate time series data with very low computational complexity. It achieves this by using an ensemble of random, non-learnable, convolutions. This method did not suit my purposes becuase it is not a deep neural network, so was not relevant to this class, but it did encourage my use of convolutions in my modification of the algorithm of Zerveas et al. [2021]. In a sense it is a modified single-layer neural network.Dempster et al. [2020] SOM-VAE emphasizes interpretability. It uses an encoder-decoder structure with a self-organizing map in the middle to represent high-dimensional time series in interpretable, low-dimensional embedding spaces. I did not use this approach because I am not using a high-dimensional time series.Fortuin et al. [2019]

3 Dataset and Features

The U.S. Environmental Protection Agency collects Continuous Emissions Monitoring System (CEMS) data on most TGUs in the U.S. which have a capacity of over 25 megawatts. I retrieved this data from the Catalyst Cooperative, a group of electricity industry stakeholders who invested in the data engineering to transform this messy public data into an easily accessible format.Selvans et al. [2022] The data I can access runs from 1995 through 2020 at hourly granularity and includes 5,977 unique TGUs. The primary purpose of this data is to track emissions of air pollutants like nitrous oxides, sulfur oxides, and fine particulate matter. However, the dataset also reports three important measures of a TGU's electrical generation operations: the fraction of the hour in which the TGU operated [measured in hours], the average power output while the TGU was operating [measured in megawatts, hereafter "MW"], and the total energy available from the fuel consumed [measured in megawatt-hours, hereafter "MWh"]. I plan to use this hourly, tri-variate time series of electricity generation from all of the TGUs for the year 2019. Using only one year will allow me to train more quickly and experiment.

For the purposes of this project, I create samples by dividing the time series for each TGU into month-long chunks. I would have preferred to use years as the unit of time, but the computational complexity of attention scales as $O(w^2)$ where w is the maximum length of the input sequence, so I chose months.Zerveas et al. [2021] This splitting of the data into TGU-months during 2019 leads

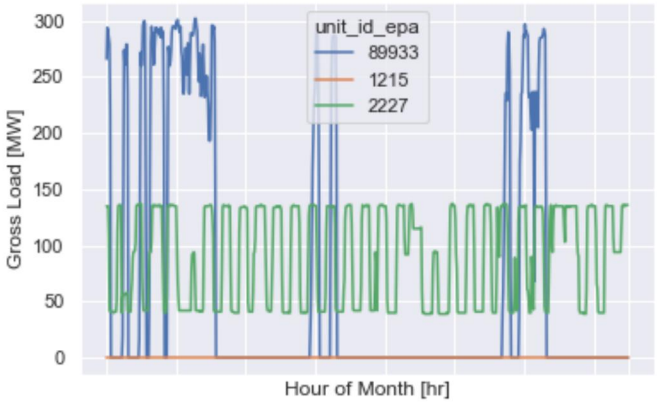

Figure 1: Comparison of Gross Load Operational Profiles for Sample Plants

me to have 50,013 samples in total. Since this is quite a few samples, I will use relatively small fractions of the total samples for my dev and test sets. Specifically, I will use 80%, 10%, and 10% of my samples fall for the train, dev, and test sets, respectively. Each of these samples, in turn, is a tri-variate time series with a length of up to 744 hours. Depending on the number of days in a month and daylight savings time, this number can vary somewhat. Figure 1 shows the average power output trajectory for three sample TGU-months. As expected, different TGUs display very different operating characteristics, making this an exciting and challenging problem.

4 Methods

4.1 Original Model

Zerveas et al. [2021] propose a transformer network, called the "Time Series Transformer" (hereafter "TSTransformer"), which only utilizes the encoder portion of the transformer architecture proposed by Vaswani et al. [2017], omitting the decoder. In place of the decoder, they propose custom output layers for each specific downstream application (e.g. regression, classification). For regression, this can be a simple linear layer. The un-supervised training proposed by Zerveas et al. [2021] occurs by means of masking portions of the input data and attempting to impute the masked values. This masking is done in a very principled way in order to encourage the network to learn relationships between variables and long-lag relationships.Zerveas et al. [2021] The loss, a mean-squared error, is computed only on the imputed values.

4.2 Modifications to Increase Speed

Zerveas et al. note that their model's computational complexity scales quadratically with the length of the input sequence. To compensate for this, they propose, but do not implement, the use of a 1-dimiensional convolutional layer before the attention layer.Zerveas et al. [2021] Since my input sequences are quite long (744 hours in a 31-day month), I decided to attempt the convolutional pre-layer architecture. I call this new model the TSTransformerConv. An alternative, simpler method, downsampling the time series to multi-hour granularity, was not acceptable for this application. Many important electric grid functions operate on a 1-hour interval, so this granularity has practical importance. Shortening the input sequence was also not an acceptable solution, since one of the goals of this work is to capture long-term time dependencies.

To achieve this, I took three main steps. First, I replaced the initial linear layer with a 1-D convolutional layer to reduce the sequence length and increase the number of channels. Second, I used a 1-D max-pooling layer to collapse the padding masks for the samples in a similar way. Then, I used the same multi-head attention layer as before in the middle of the network. After that, I used a 1-D convolution tranposed layer to map the shortened sequences back to the original sequence length.

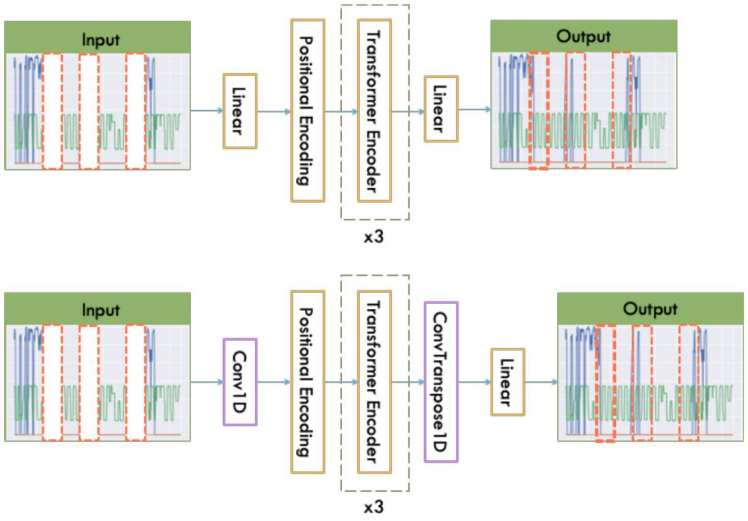

Figure 2: TSTransformer (top) and TSTransformerConv (bottom) Architectures, note the swapping of the first linear layer for a Conv1D layer and the addition of a ConvTranspose1D layer before the final linear layer

Finally, I used the same linear output layer as before. These modifications introduced two new hyperparameters, the kernel size and stride of the convolutional and transposed-convolutional layers. I decided to use the same kernel size and stride for both of these layers, since the randomized masking of the unsupervised training procedure required that the output have the same sequence length as the input.

5 Experiments/Results/Discussion

5.1 Performance metrics

For the unsupervised training phase, the loss function is the mean-squared error on the masked-and-imputed values only (see Equation 1).Zerveas et al. [2021] The set M represents the set of indices of masked values: $M = \{(t, i) : m_{(t,i)=0}\}$. Here t represents the time index and i represents the index across variables within the time-series.

$$\mathcal{L}_{MSE} = \frac{1}{|M|} \sum_{t \in M} \sum_{i \in M} (\hat{x}(t, i) - x(t, i))^2 \tag{1}$$

5.2 Hyperparameter search

I searched for the learning rate, convolutional stride, and convolutional kernel size hyperparameters. The convolutional stride and kernel size were new hyperparameters that I introduced to the model, so they needed tuning. Zerveas et al. suggested values that generally work well for some of their most important hyperparameters, so I chose to use those values without searching.Zerveas et al. [2021] I chose to take a random search strategy so that my search covered more distinct values of each potential hyperparameter. For learning rate, this was a simple uniform random sampling on the interval $[0.0001, 1]$. For convolutional kernel size and stride, it required that the two hyperparameters satisfy $\frac{n-f}{s}$ must be integer, where n is the original sequence length, f is the kernel size, and s is the stride length. This matching was essential so that the network returns the same length of sequence at the end of the deconvolution step as it took into the original convolution step. To find pairs of hyperparameters matching this critierion, I first sampled the kernel size from the even integers $\{4, 6, ..., 48\}$ Then I sampled the stride from the valid strides for that kernel size. Strides smaller than 4 would fail to achieve the compression effect I was seeking from convolution, and kernel sizes larger than 48 (two days), would probably smear the fine-grained details of the TGU behavior too much.

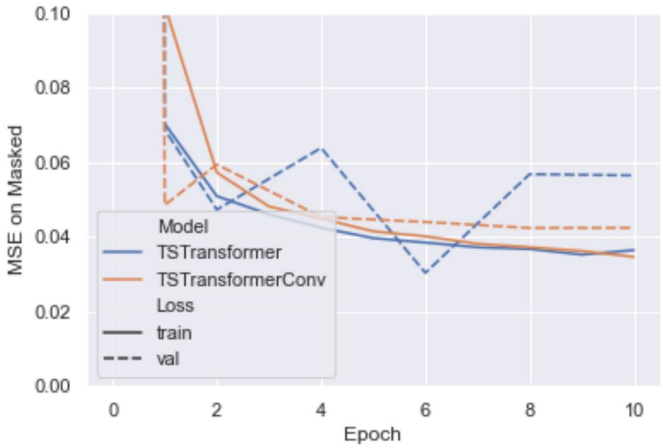

Figure 3: Comparison of Training and Validation Set Losses for the Original TSTransformer Model and My Best TSTransformerConv Model

I searched for these hyperparameters using unsupervised training based on random masking as proposed by Zerveas et al. [2021]. These random masks were re-generated each time a sample was retrieved from memory, so the masks did not stay the same across training epochs. This effect, coupled with the high length of each sample, mean that the training task for a given sample varied substantially between epochs. This may explain some of the noisy behavior in the training and validation loss trajectories (see Figure 3)

5.3 Comparison to original TSTransformer

The best model to emerge from hyperparameter tuning had a kernel size of 12 hours and a stride of 4 hours. I compared this model (TSTransformerConv) to TSTransformer trained on the exact same data. The loss was about the same for the two models (see Figure 3), taking into account the fact that the validation loss for the TSTransformer was quite noisy across epochs. However, TSTransformerConv trained 30% faster than TSTransformer. To train on the whole dataset 10 times took TSTransformer 8 hours and 49 minutes, while it took TSTransformerConv only 6 hours and 17 minutes (using one NVIDIA Tesla K80 GPU). Figure 3 illustrates that both TSTransformer and TSTransformerConv had succeeded after 10 epochs in getting most of the value out of the data, as evidenced by the flattening out of the loss curves as the epochs increase. The remaining loss must be due to bias, which could be corrected by adding flexibility to the model. For instance, an additional attention layer might help the model to improve. Alternatively, a new predictor variable, such as electricity price, could be added.

6 Conclusions and Future Work

Attention-based models can capture long-range dependencies, but they suffer from poor time complexity with long sequences. Therefore, methods of compressing sequence lengths could allow deep learning practitioners to take advantage of the benefits of attention mechanisms on long time-series. This project attempted to do just that by extending the TSTransformer with convolutional layers.Zerveas et al. [2021] The new model, TSTransformerConv, achieved the same accuracy as TSTransformer but trained 30% faster.

Future work should take at least two directions. First, researchers should add wholesale electricity price as a predictor. Intuitively, TGUs should respond very strongly to electricity prices, since their whole business is selling electricity. This data was not yet available in the dataset I used, but is scheduled to be added soon.Selvans et al. [2022] Second, researchers should explain which parts of the input sequence the model attends to. This would make the model useful for answering questions that economists pose about TGU decision-making. A research group at Harvard published code that would help get researchers started.Huang et al. [2022]

7 Code

The code for this project can be found at GitHub.

References

Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, September 2020. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-020-00701-z. URL https://link.springer.com/10.1007/s10618-020-00701-z.

Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. SOM-VAE: Interpretable Discrete Representation Learning on Time Series. *arXiv:1806.02199 [cs, stat]*, January 2019. URL http://arxiv.org/abs/1806.02199. arXiv: 1806.02199.

Austin Huang, Suraj Subramanian, Jonathan Sum, Khalid Almubarak, Stella Biderman, and Sasha Rush. The Annotated Transformer, 2022. URL http://nlp.seas.harvard.edu/annotated-transformer/.

Austin Park, Scott Jespersen, and Sally M. Benson. Finding flexibility: Data-driven parameters for grid planning and operation.

Zane Selvans, Christina Gosnell, Austen Sharpe, Steven Winter, Jan Rousik, Ethan Welty, Trenton Bush, and Bennett Norman. The Public Utility Data Liberation (PUDL) Project, March 2022. URL https://zenodo.org/record/6349337. Language: en.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. 2017. doi: 10.48550/ARXIV.1706.03762. URL https://arxiv.org/abs/1706.03762. Publisher: arXiv Version Number: 5.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, Virtual Event Singapore, August 2021. ACM. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467401. URL https://dl.acm.org/doi/10.1145/3447548.3467401.