Meter Current Transformer Polarity Detection with Deep Neural Networks

Dale Annin
CS230 – Deep Neural Networks
Stanford University
dannin@stanford.edu

***Abstract*—As homes become smarter, automation and energy consumption have become focal points of this transition. Applications of artificial intelligence (AI) has been vital in transitioning many industries. In particular, Deep Neural Networks (DNNs) are employed to determine if meter current transforms (CTs) are installed correctly with a home. With varying new energy technologies, diagnosing proper setup and configurations can be very convoluted, especially if these devices are complex. Typically, an expert such as an electrician can determine if CTs are correctly installed. Simple algorithms that look at load profiles with load and solar generated power cannot effectively determine if CTs are correct. This research looks into determining if CTs within a home is correctly installed. Evaluated is a Convolutional-Recurrent Hybrid Neural Network (CNN-RNN) baselined against a simple classification algorithm.**

Keywords—deep neural networks, load profile, meter current transformer polarity test, machine learning, solar generation, energy and power

I. Introduction

Smart home is the culmination of artificial intelligence (AI), and automation. Applying this to energy usage in the home is gaining wider adoption as solar panels, battery storage and energy efficiency devices gain popularity. Homes are typically metered at the solar, load and grid connection points within the home. Smart devices then rely on the values read from these meters to make instant and future decisions based on current, historical readings. Simple connections can be easily diagnosed if metering or connections are problematic; however, as homes become more elaborate, these simple home setups can become complex. Applying simple algorithms therefore become non-trivial and can lead to higher false positives and false negatives. Thus, Deep Neural Network (DNN) for classification is used.

Generally, if a meter is connected to a subsystem of the home where other loads, coupled with generators are collocated, determining something simple as a meter current transformers' (CTs) polarities are difficult without human intervention. Due to this, determining at scale whether a home is connected and metered properly can lead to uncertainty as these technologies scale. Consequently, metering is a very important part of a functioning smart home.

A. *Importance of Metering*

Metering is an import part of a home especially when solar generation and battery storage is involved [1]. It allows applications the ability to monitor, detect faults, forecast and predict energy usage. It works by placing voltage and current sensors at specific parts in a home, giving applications the ability to connect to the meter and query for instantaneous power, current, voltage and frequency values and accumulated energy signals over time. Applications then use this data for controls and monitoring based on a combination of the signals. If the power on a measured line exceeds a threshold, the application can trip the breaker of that line or; in a smart home sense, reduce the power consumption of a specific appliance. In a more complex setup, a solar system with a battery and multiple loads would require accurate values to determine how to react to energy consumption. For example, if solar is generating power and the home is not consuming any of it, the energy can be diverted to a home battery for future use.

II. Related Work

CT polarity detection in a smart home is fairly novel. There are a handful of companies that are now venturing into home energy automation, thus research in this field is fairly new. Current solutions require human interaction. Related work detail how to run CT Polarity Tests. These usually include running manual onsite tests or after installation tests where metering is remotely monitored or where systems send alerts on conditions. One reason for this is the inverse of the scale problem. Companies can afford to have electricians run polarity tests, usually on large sites. As small homes become smarter with solar and batteries, the problem of scale comes into play, leading to further research into this field.

III. Datasets and Features

A. *Datasets, Power and the Load Profile*

A Load Profile is an integral part of electrical engineering. Charting electrical demand over time is recorded into a load profile, which then allows analysis and insights into the particular load [2].

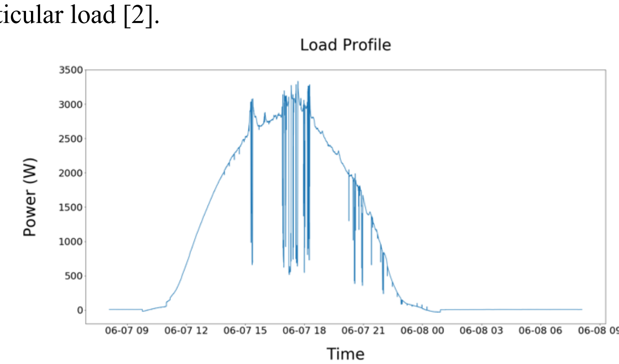

Figure 1: Solar Generation over a 24-hour period

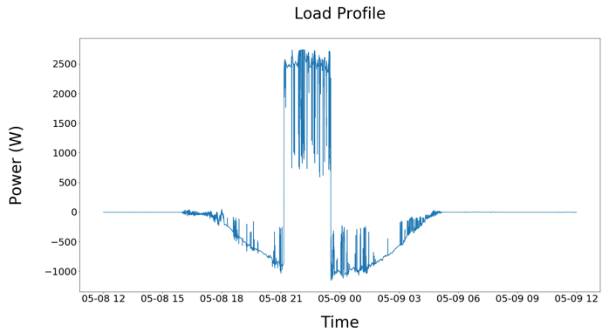

Figure 2: Solar Generation Flipped CT with Load

Power is the rate at which energy is transferred. Appliances in a household consumes power to operate. Solar Panels generate power and deliver it to appliances, stationary storage or back to the grid. A load profile is a graph of variation of the load's power over time. Load profiles are generated from power/energy readings from a meter.

Figure 1 shows an example load profile of a solar generation metered location, over a 24-hour period. For solar generation, you can see a ramp in power generated, coinciding with sun rising, and a gradual decrease at the end, with spikes in power in the middle of the day; possibly correlated with cloud cover or bad weather. Figure 2 is a representation of an incorrectly installed CTs measuring 2 solar generation sources. Our dataset is comprised of multiple load profiles with various data points.

B. *Dataset Used*

Data was gathered through a proprietary system that housed historical anonymized metering data. The data contains timestamped values that are logged at various rates per signal and on a threshold change. The data originates from systems that have generators for this investigation. For simplicity, mixed load with solar generation and purely solar generation metering will be the main focus.

C. *Features*

Table 1 shows the input futures of our dataset. Presented are timestamped power values with intervals of 15 seconds at most unless a threshold is exceeded, then an immediate value would be recorded.

TABLE I. INPUT DATA

Timestamp (milliseconds epoch)	Power (Watts)
1588975202105	2724.650024
1588975217205	2729.199951
1588975232215	2731.979980
1588975247305	2078.089966
1588975262305	2733.719971

a. Sample input data of Figure 2

IV. METHODS

A. *Data Pre-Processing*

Data may have more values logged at higher rates than 15 second intervals due to on change threshold logging. For this, load profiles are down sampled to 15-minute intervals at mean averaging. Data is also gathered in batches with the same start time. All of the data will have the same duration of 24 hours. The data will then be split up into 12-hour durations.

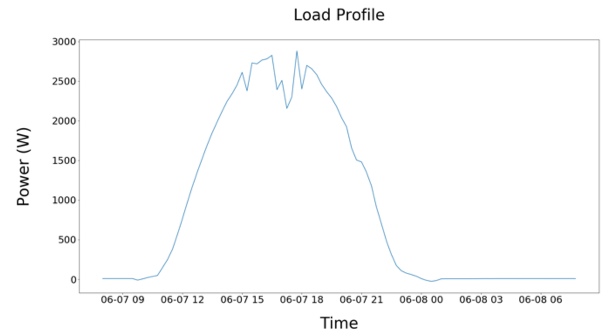

Figure 3: Down sample of Figure 1

B. *Labeling Data*

We have 2 labels: "correct polarity" (Figure 4) and "incorrect polarity" (Figure 5).

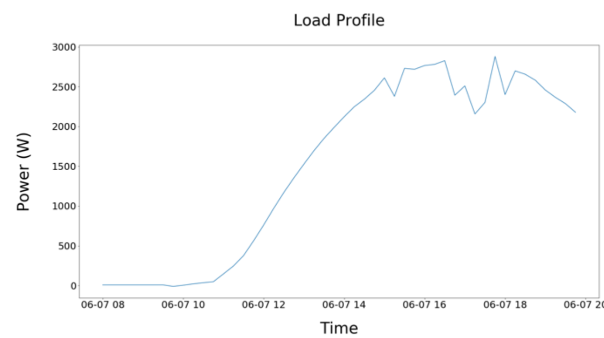

Figure 4: Correct Polarity

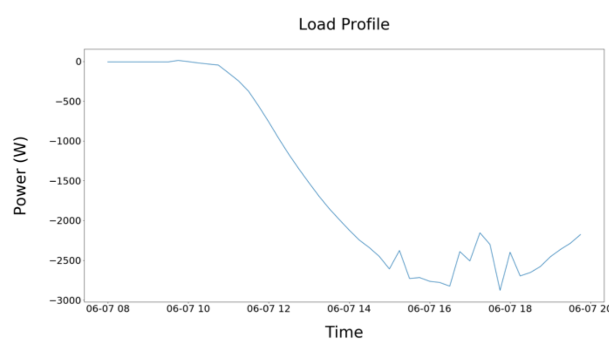

Figure 5: Incorrect Polarity

Figure 6: Unknown Polarity

C. Convolutional Recurrent Neural Network

Recognizing from the input, the sequence and size along with translation invariance needs to be accommodated in the network chosen. Convolutional Recurrent Neural Networks (CRNNs) are a perfect fit for our application. Borrowing from speech recognition research [3], as the input data is similar to audio data, Long Short-Term Memory (LSTM) are employed as the RNN to learn long-term dependencies. CNN is used to reduce the input size going into the LSTM. It also learns feature detectors that can be used in different parts of the input, which allows for parameter sharing and translational invariance. CNNs therefore, allow sparsity of connections, where the output of a CNN depends on a few inputs.

V. Experiments, Results & Discussion

In this section we discuss our results of the top models we found and our process in getting to those models via hyperparameter tuning and data normalization. We used binary cross entropy as our loss with accuracy as our metric. We divided our data into 3 partitions, train, dev and test with a split of 60%, 20% and 20%, respectively.

A. Hyperparameters

We chose accuracy as our main metric since it is suitable for binary classification tasks. Accuracy was chosen over recall because we try to maximize accuracy at the expense of getting false negatives. We experimented with SGD optimization with learning rate of 0.01and Adam optimization with learning rate of 0.01. From experimental results, Adam performed better in converging faster that SGD.

Table 2: Adam vs SGD

	Accuracy	
Epoch	Adam	SGD
1	0.6816	0.5791
2	0.7057	0.6867
3	0.8366	0.7073
4	0.9371	0.6970

Early on in the model hyperparameter tuning, we found that high batch size created high loss in our output. Using a batch size of 8 minimized that loss, and even though our final models could have used a large batch size, 8 gave great results.

We chose epochs of 50 initially but found that our models with Adam optimization converged much quickly around 10. We stopped at 10 when loss was still decreasing, and validation loss and accuracy began to deteriorate. It was also chosen to prevent overfitting and to allow time to experiment with other models.

B. Input Normalization, Batch Normilization & Dropout:

During model creation and training, we found the output of our models to be drastically wild especially in exploding loss or accuracy not increasing. We found that adding normalization of our input not only reduced overfitting, it also remedied our exploding loss and stagnant accuracy.

We split our data into train, dev and test. When our test set did poorly compared to our train and dev, adding dropout not only helped with the exploding loss, it also helped with the test set performance. We found a dropout between our CNNs and GRUs helped.

C. Experiments

Experiment 1:

For the first experiment we used 3 Conv1d with 16 filters and 5x5 kernel, each with batch normalization and ReLU activation. They fed into a GRU with 512 units, dropout layer of 0.2, then into a dense layer with 64 units. The output went through a sigmoid activated dense layer with 1 unit.

After 9 epochs:
0.0516 loss, 0.0664 dev loss, 0.2046 test loss
0.0.9847 accuracy, 0.9857 dev accuracy, 0.9035 test accuracy

Though we did well on the train and dev set, we did horrible on our test set. This meant that we overfit on the train and dev set.

Experiment 2:

Here we bumped experiment 1 convolutional layer filters up to 32, GRU units to 1024, and dropout to 0.5.

After 9 epochs:
0.0542 loss, 0.0737 dev loss, 0.0256 test loss
0.09838 accuracy, 0.9714 dev accuracy, 0.9881 test accuracy

As you can see, a bigger network and dropout increased our test set performance drastically.

Experiment 3:

Here we added a dropout after each ReLU activation in our network with 0.5 dropout.

After 9 epochs:
0.080 loss, 0.0757 dev loss, 0.0534 test loss
0.9711 accuracy, 0.9905 dev accuracy, 0.996 test accuracy

As you can see, more dropout allowed our dev and test sets to perform much better while preventing our train set from overfitting.

VI. Results & Analysis

We see that our misclassifications come down to load profiles that are hard to determine in our data set. We also have situations in our data set where values can hover around 0 for a while whilst popping up in one direction or another. As we increased the model size, it reduced overfitting our model. We also had to hand label 1000s of load profiles which could have impacted performance if we had mislabeled data. Since our data set is not that large. We tested on many other models and experiment 3 with Adam worked the best. We took preliminary error analysis and determined that we would need more data with more varying degrees of noise and CT mismatches. We also realize that more data augmentation would also help results.

VII. Conclusions and Future Work

A. Conclusions

We see that with experiment 3 gave the best results in determining accurate classification for our dev and test set.
0.080 loss, 0.0757 dev loss, 0.0534 test loss
0.9711 accuracy, 0.9905 dev accuracy, 0.996 test accuracy

We also note that CNN with GRU create a powerful DNN, mixed with Adam optimization. We experimented with other networks early on in the process and had bad to varying results, not noted in our report.

Therefore, among all our models, experiment 3 with more dropouts performed the best.

B. Future Work

Future work would be to enhance the input features. We can utilize timestamps, converted to hours and minutes; so that the DNNs can learn a correlation to the time of day. This would require scraping time-zones from each of the data points. We could also gather weather data, specifically cloudiness. With that knowledge, the DNNs can correlate dips in power to cloudiness to better predict CT polarities.

We also noticed that load profiles may have unknown polarity and thus, couldn't use it with our binary classification. In the future, we could add a third label and turn this task into a multi-classification problem. An example of unknown polarity would be when the power is always 0, or the power oscillates wildly.

New architectures have appeared that we were reluctant to dig into due to time constraints. Future work will be to try Siamese networks, ConvLSTMs, and investigate using attention layers if more data, especially weather is available.

References

[1] *The importance of current transformers (CT) in today's solar installs*, Enphasee, March 31, 2017. Accessed on: October 29, 2020. [Online]. Available: https://enphase.com/en-us/stories/importance-current-transformers-ct-today%E2%80%99s-solar-installs

[2] R. F. Chang and C. N. Lu, "Load profiling and its applications in power market," *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491)*, Toronto, Ont., 2003, pp. 974-978 Vol. 2, doi: 10.1109/PES.2003.1270442.

[3] Xiangang Li and Xihong Wu, "Long Short-Term Memory based Convolutional Recurrent Neural Networks for Large Vocabulary Speech Recognition," *2015 INTERSPEECH* Dresden, Germany. [Online]. Available: https://arxiv.org/abs/1610.03165