Non Intrusive Load Monitoring

Iby and Aladar Fleischman Faculty of Engineering Tel Aviv University
הפקולטה להנדסה ע"ש איבי ואלדר פליישמן אוניברסיטת תל-אביב

Project Number: 17-1-1-1311

By: Knaan Koosh, Ben Ohayon Advisor: Yuval Beck

Project Carried Out at Tel Aviv University

Project Goals

This project purpose is to introduce a smart and effective way to manage and monitor electricity usage in households (and other places). The main issue about NILM is the non intrusive part which mean the input of the problem can be current measured from a house during 24 hours and the output will be the disaggregation to loads used during that period. This is done without any intrusive measuring tools what makes it an interesting problem with many challenges.

Motivation

These days, every piece of equipment we use is connected to the internet, monitored and controlled for our personal use. The electrical grid is still far behind in terms of innovation. Smart power meter that can measure the total power in a house still can't disaggregate the power consumed by the different appliances without measuring the exact power each appliance use (naïve and cost inefficient). The better, more elegant and efficient way is to use smart algorithm working under the hood instead.

DFT Coefficients as Load Fingerprint

Two papers by Yuval beck and D.Srinivasan described couple of ways to distinguish loads by a unique fingerprint. This project follows these ideas and propose a new method utilize Discrete Fourier Transform decomposition to derive a 2M vector composed of M first harmonies amplitude and phase:

DFT:

$\mathrm{X_k} = \sum_0^{N-1} I_n \cdot e^{-\frac{2\pi i}{N}kn}$, where I_n is the current sampled from $I(t)$.

Amplitude and phase are calculated from DFT results:

$$amp_k = A_k = |X_k| = \sqrt{Re(X_k)^2 + Im(X_k)^2}$$
$$phase_k = \phi_k = \arg(X_k) = \arctan\left(\frac{Im(X_k)}{Re(X_k)}\right)$$

After these calculations, a fingerprint vector can be assembled:

$$FP_{I_n} = \{ A_1, A_2 \ \ldots A_M, \phi_1, \phi_2 \ \ldots \phi_M \}$$

Learning Algorithm - Deep Neural Network

Deep Neural Networks are type of Artificial Neural Networks with multiple hidden layers between the input and output layers. Neural Networks are a set of models that help us cluster and classify data according to similarities among the example inputs.

A Network element consists of an input, connection weight, and activation function.

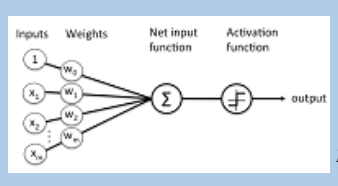


Fig 1. NN element

Feed-Forward networks are trained using methods like gradient descent. The Gradient of the error function is calculated for each weight and the weights are adjusted in order to reduce the error that the network produces.

Auto Encoders

Autoencoders are a type of neural networks that are composed of two symmetrical neural networks, one which encodes the input and the other decodes it.

Internally, Autoencoders have a hidden layer that describes a code that represents the input, our motivation of using an Autoencoder is learning latent values that describe similar inputs and later decoding them into matching classifications.

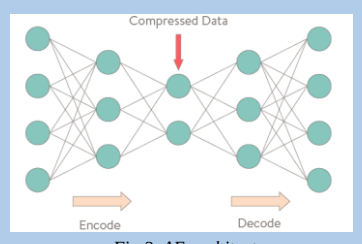
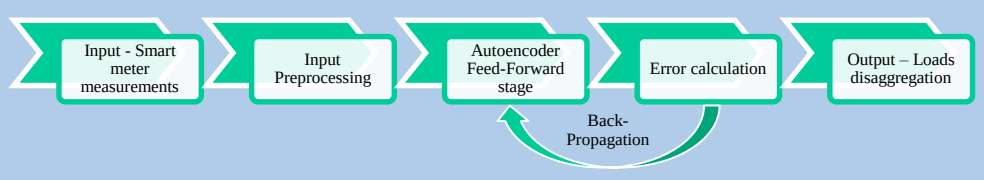


Fig 2. AE architecture

Algorithm Structure

Flow Diagram:



- Input: We receive as input electric current measurements from a smart meter with a measurement frequency of 4096Hz
- Preprocessing: Using Discrete Fourier Transformation we extract the Amplitudes and Phases of the sinusoidal waves comprising the current signal
- Feed-Forward: The transformed data is fed to the Autoencoder network and outputs disaggregation probabilities
- Error calculation and BP: The outputs and real classifications are used to calculate the error - Binary Cross Entropy:

$$\vec{L} = \sum_0^N [\boldsymbol{y_n} \cdot \log(\hat{\boldsymbol{y}}_{\boldsymbol{n}}) + (\mathbf{1} - \boldsymbol{y_n}) \cdot \log(\mathbf{1} - \hat{\boldsymbol{y}}_{\boldsymbol{n}})]$$

With gradient descent the network's weights are adjusted to minimize the error.

- Disaggregation: After the network is trained, accurate disaggregation predictions can be made for current measurements.

Simulation Results

Before dealing with real world data, we created many scenarios using generated sinusoidal signals to test our algorithm.

The generated signals are a combination of sine waves with random parameters such as frequency, amplitude and phase:

$$\mathrm{I_n} = \sum_{i=0}^{k} A_i \cdot \sin(2\pi f_i + \phi_i)$$

A Signal example:

$$\mathrm{I}(t) = 9\sin(2\pi \cdot 50 + 72°) + 5\sin(2\pi \cdot 250 - 45°) + 2\sin(2\pi \cdot 450 - 66°)$$

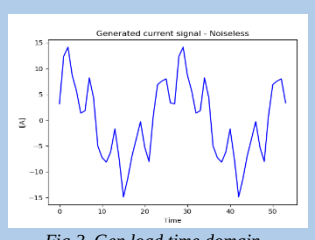


Fig 3. Gen load time domain

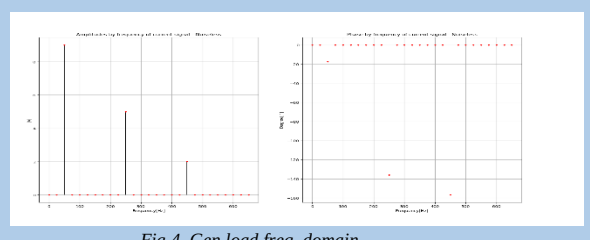


Fig 4. Gen load freq. domain

A Scenario example (5 Loads Combined):

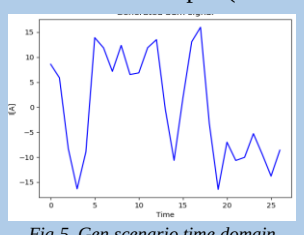


Fig 5. Gen scenario time domain

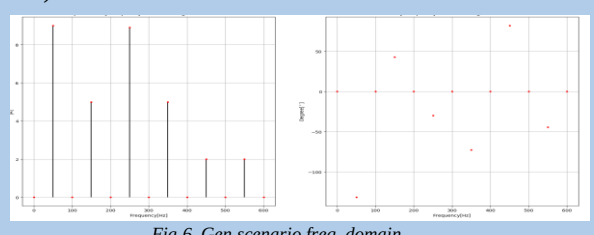


Fig 6. Gen scenario freq. domain

We disaggregated such scenarios using our algorithm, meaning the model was able to detect that the five loads were used to create the load above (Fig. 5).

To further test our model robustness, we added noise to the generated loads and increased the number of loads in the system. The following figures depict the models disaggregation

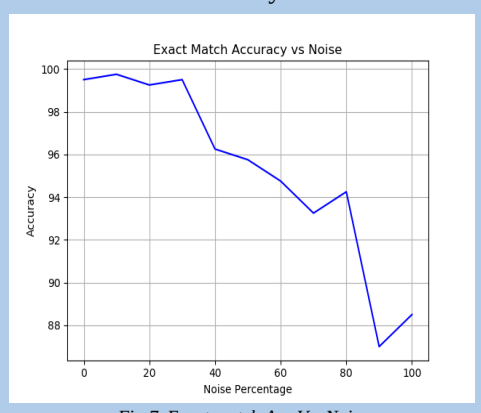


Fig 7. Exact match Acc Vs. Noise

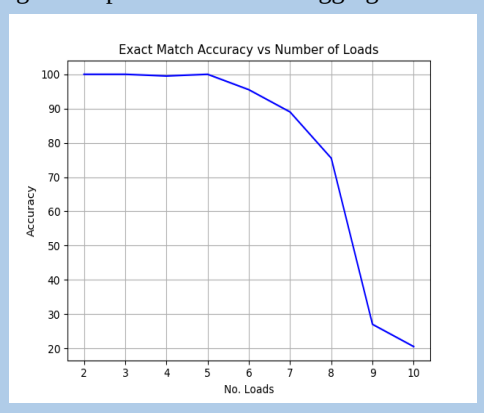


Fig 8. Exact match Acc Vs. No. loads

Lab Measurements Results

From the real-world measurements we built different scenarios comprised of these selected loads: Microwave, Toaster, Air-Conditioner, AC Motor and Lamp.

We evaluate our results with two evaluation metrics:

Exact Match: a strict metric, indicating the percentage of correctly disaggregated samples.

Hamming Loss: the hamming distance between the prediction and the ground-truth.

We compared the simulation results with the real-world results:

(note: using 5 loads with a hold-out set of 10%)

Eval. Metric (% Acc)	Simulation - noiseless	Simulation - 20% noise	Simulation - 50% noise	Real-World measurements
Exact Match	99.5	99	97	99.6
Hamming Dist.	99.9	99.8	99.3	99.92

In further evaluation, we tested our model against the number of scenarios seen and tested on all possible scenarios:

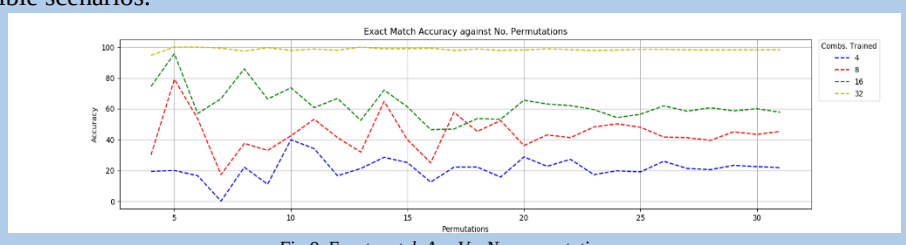


Fig 9. Exact match Acc Vs. No. permutations

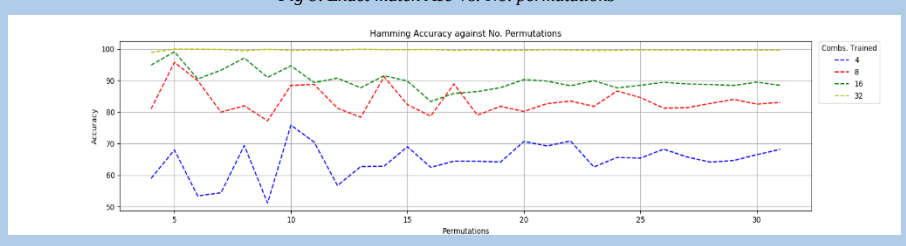


Fig 9. Hamming Loss Acc Vs. No. permutations

Conclusions

The main conclusion from the results is that the model ability to disaggregate relies on the amount and quality of the data it consumes. The simulated and real-world results indicated that the model can handle high amount of noise yet scaling to a bigger number of appliances may require changes in the network architecture. Furthermore, the model was unable to generalize well, meaning it could not predict a scenario that it didn't encounter.

Possible improvements:

- Using Recurrent-NN to account for temporal features.
- Exploring different loss functions for multi-label classification.