NILM Power Disaggregation via Artificial Neural Networks

Roberto Martinez, Daniel Pacheco, and Dane Robergs

Institute for Complex Additive Systems Analysis (ICASA)

New Mexico Institute of Mining and Technology (NMT)

***Abstract*— Widespread use of electronic systems has led to a need for better monitoring systems. NILM allows users to monitor their devices for excessive consumption, faults, or security lapses. To understand the large amount of data from NILM systems, a power analysis technique must be employed. Current methods involve creating an analysis algorithm to extract certain features from the measurements and analyze them based on prior knowledge of how the devices behave. These methods are often time-consuming to develop as in-depth knowledge of how the device performs is required. Convolutional Neural Networks (CNN) can bypass this as the algorithm can learn such features on its own. As described in this paper, a CNN was developed for use as a NILM analysis tool. The CNN is currently in the early stages of its development, but results from preliminary training are as expected. Network performance is showing the network is, in fact, learning a device's transient event to predict its state accurately.**

Keywords*— Non-Intrusive Load Monitoring, Artificial Neural Networks, Convolutional Neural Networks, transient events, preprocessing.

I. Introduction

As the production of consumer electronic devices increases yearly, it is inevitable that homeowners, business owners, and many others will continue to lose familiarity with the devices whose activity composes the majority of their electric utility bill. Appliances with excessive power usage due to defective components, over-use, or unnecessary activity will go unnoticed among the plethora of other electronic devices, causing such problems never to be noticed or addressed. A direct but naive solution to this issue would be to monitor individual appliance power consumption; a method that requires an expensive investment to both procure and install the required sensors for each independent device. This solution is known as Intrusive Load Monitoring (ILM) and is practically untenable for any scalable implementation.

Non-Intrusive Load Monitoring (NILM) presents an excellent alternative to this solution by providing the same information at the cost of a single sensor with moderate signal processing capabilities. This method is theoretically capable of identifying and "learning" new devices and normal patterns of appliance usage in a particular system. Apart from naturally identifying faulty or failing components in the system through methods of predictive maintenance, this behavioral modeling has the potential to identify cybersecurity threats by detecting irregularities in the behavior of a connected device.

The purpose of this project is to develop an efficient and effective NILM tool by employing robust machine learning techniques. Specifically, an Artificial Neural Network (ANN) will be constructed to analyze active electrical power consumption at approximately 12 kHz. The ANN will be expected to perform feature extraction, classification, and state identification by training it on a suite of distinguishable devices and operating states. The network will be constructed in Python using the neural network package Keras. Documentation and information for Keras are given in [4].

II. Background

The concept of an ANN was first coined by Warren McCulloch and Walter Pitts [8] in 1943 as a technique for analyzing large datasets. ANNs generally model their behavior after biological neurons in the human brain where they "learn" by adjusting the weights of connections between simulated neurons. These weights control the amplitude of incoming signals to the neurons, and as such, signify the importance of the information passing through them. Fig. 1 illustrates the structural and mathematical representation of a simulated neuron.

Each neuron in an ANN takes in a family of inputs and amplifies them based on the weight of each input path. An activation function is then applied to the net input for a given neuron. If the activation exceeds a threshold (determined on a per-neuron bias) then the neuron "fires," and the activation value is sent forward to the next set of neurons in the network. A collection of adjacent non-interacting neurons is called a layer with each neuron in the layer passing their information forward to the next layer. In analogy to the biological system, unused (inactive) neurons will die off over time, thereby reducing the complexity of the network and better refining it for its assigned task. A traditional neural network is created by stacking several layers of varying size as shown in Fig. 1.

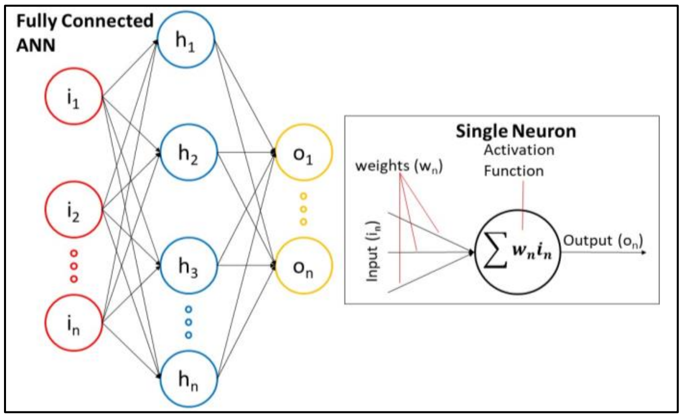

Fig. 1. A fully connected network highlighting the processes in a single perceptron

III. Neural Network Structure

Convolution Neural Networks build off the general ANN by adding an efficient preprocessing network. The combined convolutional layers create their own network which is then added to the input layer of an ANN. The layers create a receptive field (or feature map) where different regions indicate different types (or sources) of stimuli. A CNN was constructed using two convolutional layers, three hidden layers, and a final output layer. The following sections will discuss in detail each layer within the network.

A. Inputs

It is important to note that, up until the introduction of ANN methods, NILM experiments were composed of discrete processing stages (including transient detection, extraction, processing, and classification) that were independently accessible and configurable. Alternatively, neural networks operate utilizing tensors in which physical measurements are directly mapped to a feature space. The accuracy of the resulting classification is dependent on the cumulative training of the entire network whose constituent elements are not independently configurable.

Our network was created to accept voltage and current measurements from a power source. The measurements were taken from a single point, such as an electrical breaker, to satisfy the basic NILM concepts. Feeding the raw data into the network minimizes any preprocessing that would need to be done to analyze other common NILM data, such as calculating real or reactive power.

B. Feature Extraction

Feature extraction is a process of dimensionality reduction: a process where large datasets are reduced to their “most important” features. The process aids in decreasing the amount of time spent learning repeated or non-informative information. Instead of using manually tuned, user-defined parameters to extract individual transient patterns from surrounding noise, the convolutional layers at the beginning of the network can learn how a transient event is represented in a unique feature space defined by the training data. This representation will then be used to classify the device without requiring substantial manual involvement or interpretation from the user. Individual layers comprising feature extraction will be discussed further below.

1) Convolutional 1D: A Convolutional 1D layer takes input data of 3D shape and creates feature maps for each filter. This is done via the dot product of the kernel (weight) and input (comprised of the width and height) matrices. The feature maps are stacked along the depth of the input tensor to create the convolutional layer. Once network training is complete, the filters inside the convolution layer will slide over the input matrix until a specific feature map is activated [21]. This activation is the result of the network recognizing a feature from training. For the network 1000 feature maps were created with a kernel size of 3 and an output shape equal to the input, ensuring that the network could accurately represent the transient events.

2) Max Pooling 1D: A Max Pooling 1D layer is a method of non-linear down sampling where an N x N matrix is separated into sections of size M x M. The max value from the section is extracted and stored in a matrix also of size M x M. For the network the data was downsampled by a factor of 2 or half its input size.

C. Long Short-Term Memory Layers

Long Short-Term Memory (LSTM) layers remember previous events to help the network predict current and future events. An LSTM is a modification to a Recurrent Neural Network (RNN) where information regarding an event persists within the network but solves the long-term dependency problems commonly seen in RNNs (the information needed to predict an event occurred too far in the past) [2]. An advantage of an LSTM layer is it gives the network the ability to learn from previous events to classify current ones. Instead of classifying a device and discarding the information the network can remember an event by feeding the output data back into the layer input. For example, a coffee pot is a multi-state device with a long initial activation time (coffee brewing) followed by a series of shorter reheating events until the device is deactivated. A traditional neural network would detect and classify the brewing transient, discard the information, then detect and classify the reheating transients individually. An LSTM layer, however, would remember that a series of reheating transients occur shortly after a brewing transient; this would aid in better classification accuracy. The fully connected hidden layers are comprised of these LSTM layers, and the parameters for each are discussed more in-depth in the following sections. Once the individual layers were initialized, they were compiled to form a single model. The compiled network is illustrated in Fig. 2.

1) LSTM 1 and 2: The number of neurons for each of the hidden layers is proportionally related to the accuracy of the network; based on this and that computational speed is increased if the values are a factor of 8, the layer values were chosen as 608 and 352 neurons, respectively. These layers contained dropout, bias, and return sequence parameters. Dropout decreases the potential for overfitting by removing neurons from the network on a temporary basis, at a preset rate (A 0.2 dropout rate for the network). Bias is also a means of reducing overfitting as it randomly shifts the origin of the activation functions for each layer in the network. The return sequences parameter ensured the full 3D shape of the input tensor was maintained in the outputs. An activation function of Rectified Linear Unit (ReLU) was used to decrease the effects of vanishing gradient.

2) LSTM 3: The final hidden layer with 208 neurons and a ReLU activation function. The “return sequences” parameter was set to false, thus, collapsing the 3D input tensor into a 2D tensor. The outputs from this layer will be stored and fed into an output layer.

D. Classification

Currently, the network can predict the device undergoing a transient event. This is done by feeding the 2D outputs from

LSTM 3 into a Dense layer with a sigmoid activation function (ensures all the data is between 0 and 1). A Dense layer is a traditional feedforward neural network layer in which an activation function is applied to the sum of the weights multiplied by the inputs. The network was trained to detect the transient events of 161 devices; a process which will be discussed further in the following section.

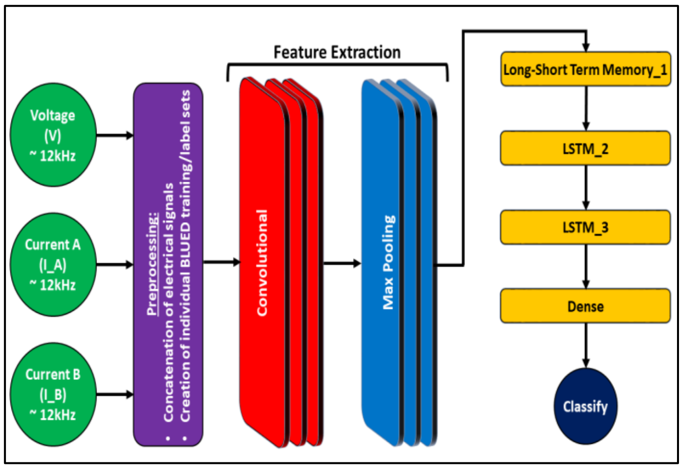

Fig. 2. Network structure outputs after the layers were compiled.

IV. Network Training and Evaluation

A training dataset was constructed by extracting the transient events from the BLUED database. The database is comprised of 16 location files with each file containing around 400 continuous data files of voltage and current. Each location file provides a list of the time at which the events occur. This time was converted from 24-hour to seconds. The files were individually loaded and searched for the event time. A script file was created in MATLAB to perform the extraction of all location files.

To train the network, the layers were combined in sequence with each other. The network was trained and evaluated using the Keras functions "fit" and "evaluate." The training set is comprised of 16 individual files of extracted transient events. To create a test and validation dataset, 7 of the 16 files were randomly chosen and compiled. The purpose of the test dataset is to check network training for overfitting by comparing the final prediction against a similar dataset. If the network is unable to predict the output from both sets accurately, then the network is retrained. A validation dataset is used after training to test the overall accuracy of the network. Details regarding the loss and optimization functions can be found below.

1) Loss Function: The purpose of the loss function is to determine the current fitness of the network. During each iteration of network training, if the current loss at the end of the network is above a certain threshold, then the weights of each neuron are changed, and the loss is calculated again. This is done until the network produces a loss small enough to not underfit, but large enough to not overfit (0.05-0.2 is a good range). The loss function used for the network was binary cross-entropy, which decreases the likelihood of a decreasing learning rate [5].

2) Optimization Function: An optimization function was used to minimize the loss function; whose global minimum defines the ideal operating point of the classifier. The optimization function chosen for the network was Adaptive Moment Estimation (Adam). Adam is an adaptive learning algorithm with the ability to diminish the effects of "vanishing learning rate, slow convergence or high variance in the parameter updates which leads to fluctuating loss function" as stated in [5]. For network evaluation, the test and validation datasets were used.

To test that the network was operating as expected, it was trained on a single training file. The simple testing allowed the network to be monitored more closely as fewer values were being inputted into it. The training process was set to plot the loss per epoch. As can be seen in Fig. 3, the network loss is generally decreasing as the number of epochs increases. The loss value stabilizes between 0 and 0.005 which indicates the network is overfitting and that a global minimum has been reached. Overfitting was expected in this initial testing as the network was trained on a single training file with no comparison test set being compared in order to simplify the process.

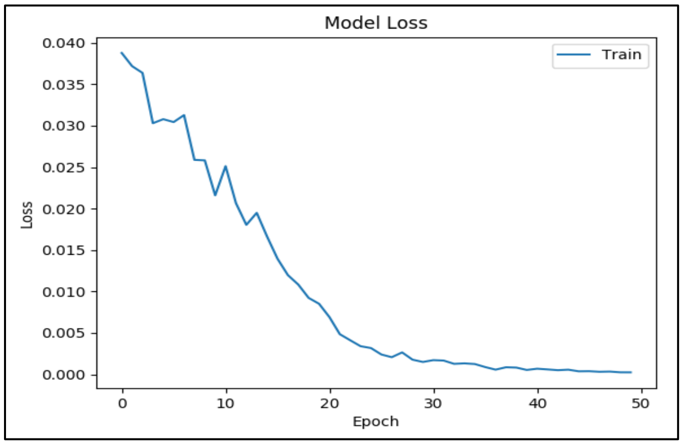

Fig. 3. Model loss as a function of epoch.

V. Data Collection

To test the network's ability to analyze data from various sources, a data collection experiment will be performed. From this, we will generate our own dataset to test and expand the capability of the neural network. The experimental setup will be described below.

The voltage and current usage will be measured via a circuit involving an AC-AC adapter, a Current Transformer, and a PC sound card with multiple line-in ports. As illustrated in Fig. 4, there will be two inputs from the household power to the circuit, one for the voltage measurement and one for the current measurement. The AC-AC adapter will transfer voltages from 120V to 12V, and a voltage divider will lower the voltage to within the sound card tolerances at the left ADC which will measure voltage. The Current Transformer which will lower the current to a ratio of 10000:1 has a burden resistor allowing the line-in at the right ADC to safely measure current. The sound card will collect the data and MATLAB's audio recorder functions will be able to collect and interpret the raw data.

Modifications were made from the original setup described in [12] as a UK power system utilizes 240V instead of 120V as in the US. This included changing the resistor values to account for the smaller voltage, and increasing the voltage passed through the sound card to allow for a cleaner signal.

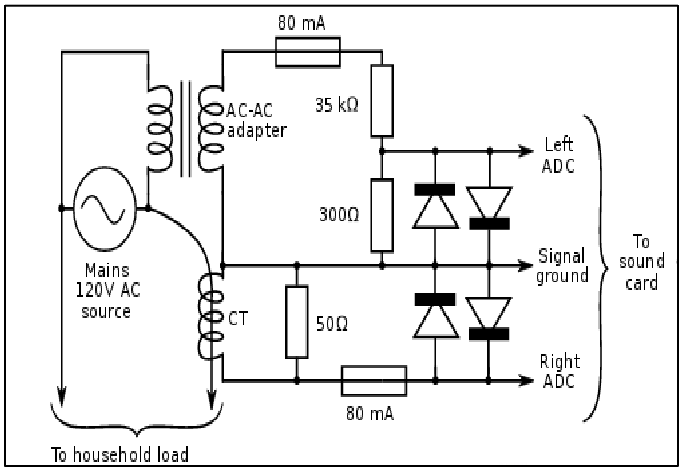

Fig. 4. Experimental setup for in-house data collection [12]

VI. Advantages Over Other Methods

CNNs are typically applied to image classification problems because of their superior capabilities in pattern recognition and classification. This gives them an advantage in NILM analysis over other classification algorithms as the network can learn which features best represent a transient event. Other methods require a detailed analysis of what these features would be before developing the analysis technique. For example, a previous NILM experiment performed by the authors dealt with classifying devices based on their real power activation and deactivation transients. Determining the criteria to detect when an event occurred required an in-depth study into the appliance's behavior. The method relied on monitoring the signal for known changes in the mean, standard deviation, skewness, and kurtosis. Once the criteria were met, the transient event could be extracted and analyzed. The method was very susceptible to noise variations as the signal noise would sometimes trigger the event criteria, leading to misclassifications. Another method where such in-depth knowledge was required can be seen in [22].

VII. Conclusion

In this paper, a machine learning algorithm for NILM was discussed. Such an algorithm would be capable of analyzing the electrical signals to predict the devices being used. CNNs eliminate the need to predetermine the features that compose a device's transient event. This shifts the workload from individually analyzing a device transient and creating a device-specific solution to creating a single, functioning CNN.

Future work would be to design a single NILM system to perform data collection, storage, and analysis. Such a system would be able to detect unwanted access on a network and perform predictive maintenance. Also, adapting the network to perform real-time analysis on signals would be crucial for the advancement of a single NILM system. CNNs ability to function in a big data environment allows their applications to extend beyond a household monitoring system.

References

[1] Hart, G. (1992). Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12), pp.1870-1891.

[2] "Understanding LSTM Networks -- colah's blog", *Colah.github.io*, 2017. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 31- Jul- 2017].

[3] Abubakar, I., Khalid, S., Mustafa, M., Shareef, H. and Mustapha, M. (2016). RECENT APPROACHES AND APPLICATIONS OF NON-INTRUSIVE LOAD MONITORING. *ARPN Journal of Engineering and Applied Sciences*, 11(7), pp.4609-4618.

[4] "Keras Documentation", *Keras.io*, 2017. [Online]. Available: https://keras.io/. [Accessed: 31- Jul- 2017].

[5] "Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent", *Medium*, 2017. [Online]. Available: https://medium.com/towards-data-science/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f. [Accessed: 01- Aug- 2017].

[6] Haiping Lu, Plataniotis, K. and Venetsanopoulos, A. (2008). MPCA: Multilinear Principal Component Analysis of Tensor Objects. *IEEE Transactions on Neural Networks*, 19(1), pp.18-39.

[7] Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. [online] Deeplearningbook.org. Available at: http://www.deeplearningbook.org/ [Accessed 10 Jul. 2017].

[8] "Artificial neural networks: fundamentals, computing, design, and application - ScienceDirect", *Sciencedirect.com*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167701200002013. [Accessed: 01- Aug- 2017].

[9] Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P. and Bengio, S. (2010). Why Does Unsupervised Pre-training Help Deep Learning?. *Journal of Machine Learning Research 11*, (11), pp.625-660.

[10] M. Nielsen, "Neural Networks and Deep Learning", *Neuralnetworksanddeeplearning.com*, 2017. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap3.html. [Accessed: 01- Aug- 2017].

[11] (2017). *Log Analog Input Data to a File Using NI Devices.* Retrieved from https://www.mathworks.com/help/daq/examples/log-analog-input-data-to-a-file-using-ni-devices.html

[12] J. Kelly and W. Knottenbelt. "The U.K. dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," Department of Computing, Imperial College London, London, SW7 2RH, UK.

[13] W. Hays. "Exploring Current Transformer Applications," BH Electronics, Marshall, Minnesota, June 2002. Retrieved from https://www.power electronics.com/content/exploring-current-transformer-applications

[14] (2017). *Delta 66 Manual.* Retrieved from https://www.americanmusical .com/ItemFiles/Manual/maudio/delta66_manual.pdf

[15] He, K. (2016). Deep Residual Networks.

[16] Hoyo-Montano, J., Pereyda-Pierre, C., Tarin-Fontes, J. and Leon-Ortega, J. (2016). Overview of Non-Intrusive Load Monitoring. *13th International Conference on Power Electronics (CIEP)*, pp.221-226.

[17] Iwayemi, A. (2016). Non-intrusive Load Monitoring and Demand Response for Residential Energy Management. ProQuest.

[18] Jain, A., Mao and Mohiuddin, K. (1996). Artificial neural networks: a tutorial. *Computer*, 29(3), pp.31-44.

[19] Le, Q. (2015). A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks.

[20] Reznik, A. and Dziuba, D. (2009). Dynamic Associative Memory, Based on Open Recurrent Neural Network. In: *Proceedings of International Joint Conference on Neural Networks*. pp.2657-2663.

[21] Cs231n.github.io. (2018). *CS231n Convolutional Neural Networks for Visual Recognition*. [online] Available at: http://cs231n.github.io /convolutional-networks/ [Accessed 7 Feb. 2018].

[22] Lange, H. and Berges, M. (2016). The Neural Energy Decoder: Energy Disaggregation by Combining Binary Subcomponents. In: *NILM Workshop*. [online] Available at: http://nilmworkshop.org/2016/ proceedings/Paper_ID19.pdf [Accessed 7 Feb. 2018].