Transmission Branch Event Detection and Identification in Energy Control Centers

Jackson Murrin
Holcombe Department of Electrical and Computer Engineering
Clemson, South Carolina
jmurrin@clemson.edu

James Nguyen
Holcombe Department of Electrical and Computer Engineering
Clemson, South Carolina
jhn@clemson.edu

***Abstract*—With the implementation of the smart grid being seen all over the world, outage detection and avoidance is one of the key goals. A more reliable and self-healing grid is a common goal of the utility providers and the customers at the other end. This has caused researchers and engineers to develop methods which can help detect and alert operators of potential problems from real time PMU data. Our project looks at current and voltage data collected from various points along the transmission lines from generation to PMUs, finally to the loads at the end of the lines. This data is then used to detect possible outages within the transmission lines using our detection algorithm. Neural network learning is used to detect these system events. This outage detection is finally conveyed to power station operators in the visualization tool created for this project. Goals for this project include getting detection of grid events as close to 100% as possible and creating a accurate visualization tool which shows real time status of the transmission lines.**

***Index Terms*– Feedforward Network, Neural Networks, Outage Detection, Smart Grid**

I. Introduction

This project will focus on detection of events within the transmission branches of a smart grid. These events are simulations of what it would look like if there were to be an outage somewhere in the physical grid. A diagram of the simulated power grid can be seen further into the paper, but it includes both traditional and photovoltaic power generation, transmission lines to substations, phasor measurement units and then transmission lines leading to seven different load values at the end of the lines. Voltage and current data was given from the various branch breakers along the transmission lines. Each transmission line was also up to the n-1 standard allowing for increased grid healing.

To start developing a method for an intelligent learning algorithm, many papers were studied to try and gain an understanding for the research in this field that has been already completed. Within this introduction section, a summary of a few of these sources can be read. Lots of useful information was gained on different algorithms tested using neural networks, the importance of PMU data collection and different ways of preventing an outage before it happens.

Matlab was found to be a very useful tool in the implementation of an already existing neural network. Building upon their previously built toolbox of an intelligently learning neural network, the provided input and output files for branch data were able to be analyzed properly. Matlab also provided the tools necessary to build a graphical user interface which could tell the operator of the current status of the power grid. One of the most important concepts of the neural network was the implementation of the feedforward neural networks. The concept can be found described below from a literature review. The other major topic which is used is the levenberg-marquardt algorithm for neural network training. This concept is also introduced in a literature review found below. Building upon this already created toolbox allows us to focus on optimizing performance of the results.

The first journal article [1] emphasizes that branch event detection in the transmission network is crucial to develop the ‘self-healing’ aspect of the grid and the goal is to design a fast system to detect branch events using PMU measurements in a complex, nonlinear system. Artificial Neural Network (ANN) is one popular technique under AI that is explored with this study. Specifically, it is efficient for small systems and with a decentralized grid many smaller systems can work together to form our large power system. The Cellular Computational Network (CCN) is one of these smaller architectures. This study uses a feed-forward multilayer perceptron neural network which uses the inputs to infer the output based on intermediate computations. The neural networks were trained by supervised learning method. This study analyzes PMU data using two methods: 1) A single multilayer perceptron (MLP) neural network and 2) A cellular multilayer perceptron (CMLP) neural network. Both approaches were successful at identifying network branch events, but MLP identification was better, but CMLP had a faster computational time for each cell. Computation burden was higher for the MLP

method which supported the conclusion that CMLP was more useful and practical during this experiment. The study suggests extending its experimentation to larger power systems, to investigate more efficient methods of using AI, and to expand research beyond just the (N-1) contingency level.

For the second literature review article [2], the concept of feedforward neural networks was explored. When creating a neural network getting the appropriate size is very important. The layers within the network and nodes per layer and the number of connections are a determining factor of this problem. A nonlinear mapping function u=G(x) is used to try to learn to associate input patterns to output patterns. This is why accurate training data for both inputs and outputs are needed for the neural network to develop an accurate mapping function. Generally the number of nodes can be found for the input and output layers through the dimensionality of the problem, but the number of hidden layers is more complex. Nielsen has shown that a network with two hidden layers can approximate all nonlinear functions and get an accurate decision region for problems. Later on Cybenko showed that a close approximation to nonlinear decision making can be done with just a single hidden layer. Hornik and Stinchombe built upon this research to determine that single hidden layer feedforward network can accurately approximate virtually any function of interest to the desired degree of accuracy. The feedforward neural network is considered the first and the simplest type of neural network developed. It only operates in a forward direction moving from input to hidden layer to output with no cycles being formed. Networks are better when as small and simple as possible. You can always add onto the network if needed but it is harder to deallocate the resources which are unused. It was also found that bigger networks perform poor mappings when given limited training data. The paper then steps through different pruning methods, constructive methods, sensitivity based methods and error function modification to better understand the logic behind feedforward neural networks and how they grow and shrink. The author suggests a combination of constructive and pruning methods but that starting small is generally the better start.

In the third literature review, the Levenberg-Marquardt algorithm was introduced as the preferred training method with a neural network. This algorithm is a damped least squares approach to solve non linear least squares problems. As explained in the previous literature review, a feedforward neural network needs to solve for a non linear mapping function. Therefore, the most popular method for doing this is with the Levenberg-Marquardt algorithm. It helps to optimize least square curve fitting, which a feedforward network wants the least amount of polynomial terms as possible to simplify the curve. There are two main concerns which the article raises as drawbacks to using this algorithm and that is it is not biologically feasible and it is both time and memory consuming. For the application used within the problem statement, none of these concerns are of major impedance to results. The paper then discusses how Matlab implements this Levenberg- Marquardt algorithm in its neural network toolbox. The rest of the paper discusses alternative methods to using this algorithm and how though some might perform better, they are much more complex, not easily implemented and more need specific versus a general algorithm for most all neural network problems. With it being the most widely used and simple algorithm at solving for a non-linear least squares equation, the group is confident in its ability to work for the applications of the smart grid project.

II. Methodology

To analyze the amount of data coming into the system, a versatile learning algorithm is needed. Therefore, a neural network was decided upon to model and create this knowledge base. A neural network is an adaptive system that learns by using interconnected nodes or neurons in a layered structure that resembles a human brain [4]. They can learn from data and therefore recognize patterns, classify data and predict future events. To create an efficient neural network, our group decided upon using the existing matlab toolbox for learning. With this existing groundwork, it allowed us to refine the tools for our specific purposes.

Matlab uses a feedforward neural network set up. This is explained in detail from one of the literature reviews found in the introduction section of the paper. To give a brief summary, a feedforward neural network is one that does not form a loop or cycle and only moves in one direction. It includes an input layer, hidden layers and an output layer. It is considered to be one of the most basic but versatile neural networks.

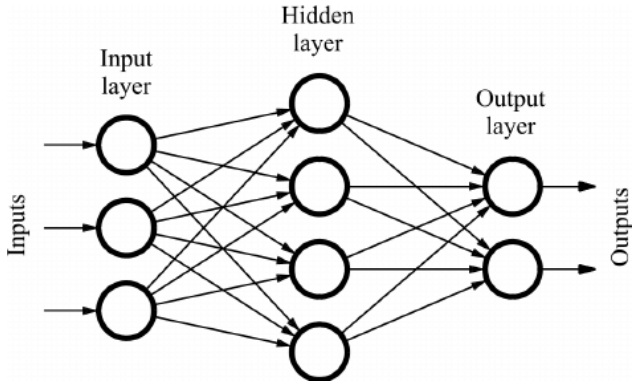

Figure 1: Feedforward Neural Network

A feedforward neural network uses the Levenberg-Marquardt algorithm to help find the non linear least squares equation for the data set provided. This algorithm can be further explained in the introduction section literature review previously in the paper. To summarize, it is the an algorithm which helps to optimize the least squares curve fitting built from the training data, to then be used on the real time power data coming from the transmission lines. It is considered to be the most widely used neural network optimization approach and adaptable to most all problems that arent of a biological topic, which the power grid is

obviously not. As seen in the equation flow chart below, this algorithm is very complex and without the help of the existing matlab toolbox, not obtainable by the common engineer.

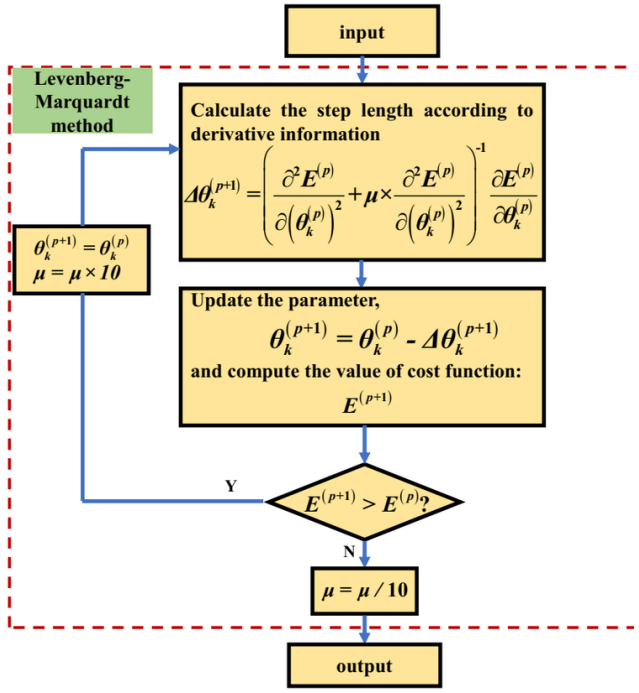

Figure 2: Levenberg-Marquardt Flowchart

III. Experimental Design

The test system being used for this project can be seen modeled below in figure 3. This is a simplified transmission branch diagram which has a few key similarities of a typical power grid but simplified to an extent for this project. It includes four power generation plants, two photovoltaic generation areas, sixteen phasor measurement units, seven different loads at the end of the lines and follows an n-1 standard for line transmission allowing for ease of not allowing for a line outage to cripple the whole grid. With the modern power grid relying so heavily on real time data and lots of it, new ways of analyzing and filtering this data need to be created. For this project nineteen different points along the transmission line are actively creating data which is to be analyzed. This comes in the form of currents and voltages in phasor form.

The training data files are for the inputs and outputs of the system and these are used to train the neural network about trends and have it learn the typical grid performance. This nonlinear equation which is developed using the Levenberg-Marquardt algorithm is then used against the testing data files.

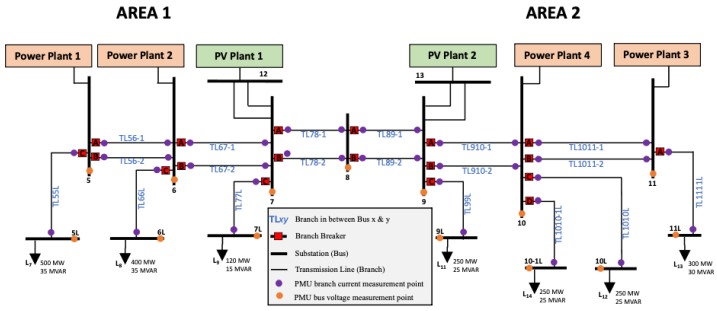

Figure 3: Transmission Line Diagram

Within the matlab code written for the analysis of the data, the first step was to read in the training input and output files and format the data in a way to correctly use later on. Next, the neural network was set up. This was done by calling the "feedforwardnet" function and then using the "train()" function with the input and output training data as parameters within it. The neural network was then simulated using the testing input file which was read in. A simple for loop was coded to set 1 or 0 values for a certain threshold within the read values in the data. Finally, output data was analyzed and an error count and percentage accuracy was calculated within the code to be printed. Overall, with the existing matlab toolbox, creating a functioning and highly accurate feedforward neural network was not a difficult task.

For the final piece of learning, the group opted to create a distributed architecture neural network. So, this involved setting up 19 different neural networks, one for each branch of the grid. As will be discussed in the next section, this helped increase accuracy of the model. It is not too complex in setting up and as will be seen, increased the accuracy of detecting events in the grid. As seen in figure 4 below, this is an example diagram of what a distributed neural network looks like. It can be observed that the server has two way communication with each of the different nodes and weights for the neural networks but none of them intercommunicate, as is the case with our code which creates 19 different neural networks for each transmission line, without speaking to each other. The server is the main point of contact.

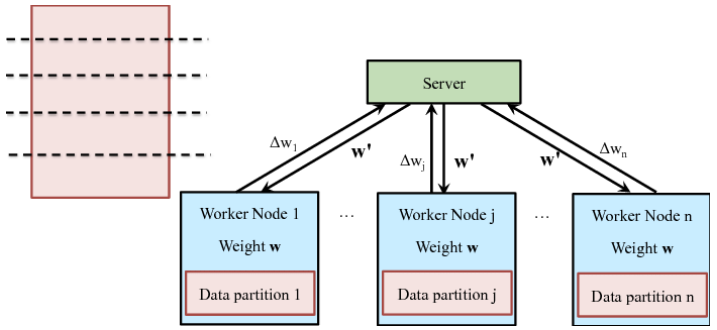

Figure 4: Distributed Neural Network

For the final part of the project, a data visualization had to be created which allows for the operator of the power station to get real time updates on the status of the various lines within the grid. Since the data analysis code was completed within matlab and a prior matlab operation usage is had, the group decided matlab to be the best choice to create the visualization in. Matlab has a graphical user interface tool to create apps for the desktop. The figure 5, seen below, is the outcome of the created user interface. It has the original transmission line image which was provided at the beginning of the project but gets green or red lines imposed over the top of the transmission lines when they are either in proper operation or not working. As seen in the figure, a start button was included on the GUI as to tell the program when to start showing the real time updates on the transmission lines.

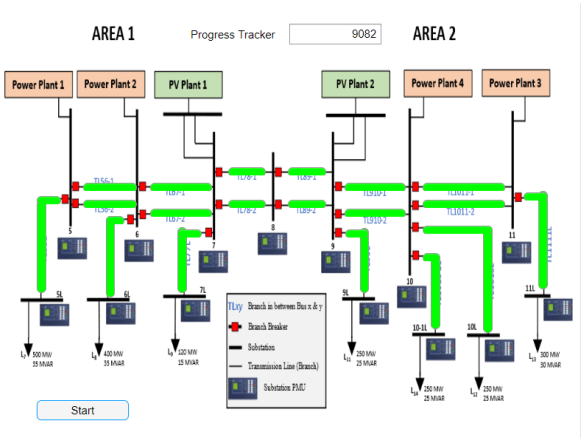

Figure 5: Visualization Tool Created

IV. Results and Discussion

The results of our experiment led to a near 100% accuracy for detecting branch events. The Matlab toolbox for neural networks has twelve different training methods. When testing different training methods, the things considered most were percent accuracy of detecting branch events and the amount of time to train the neural network. Our method imports all of the input data, isolates the current data in the input data, and then uses a neural network to find the relationship between the current data and the activity of the branches (online or offline). Overall, the Levenberg-Marquardt method seemed to be the most effective since it had the second highest accuracy but with a reasonable training time. Other methods had very short training time, but accuracy was valued over training time. In the case of Bayesian Regularization, the accuracy was higher but the training time was very high. All of the testing for the neural networks were performed with a hidden layer size of 10.

To achieve even higher accuracy, we created a neural network for each branch for the final part of the project. Training 19 (1 for each branch) levenberg-marquardt neural networks, which pushed the accuracy from 98% to 99.8% on average. Each neural network had a reduced layer size of 3 to speed up the training process. Though that only seems like a small improvement of only 1.8%, this is a vast improvement since when being used on the grid, consumers can't afford for the accuracy to not work 2% of the time. So, this is a big improvement getting the accuracy within 0.2% of perfection. Also, training 19 neural networks separately significantly decreased the amount of time needed for training. Using 19 neural networks allowed the data to be split up so that the patterns between the inputs and outputs was much easier to identify with greater accuracy and higher speed.

Based on the results gathered and the ease of creating a levenberg-marquardt neural network for each branch, our group would recommend this method for application in the real life power grid to help detect events. This distributed architecture design is a proven high accuracy approach and allows for the neural network learning to occur for every branch. We feel as though this is a very attainable piece of code for any utility to apply to their grid and further refine it using their personal specifications to get it even closer to 100%.

Training Method	% Accuracy	Training Time
Levenberg-Marquardt	98%	1:16
Gradient Descent	91.40%	0:08
One Step Secant	97.50%	0:11
Scaled Conjugate Gradient	97.00%	0:03
Fletcher-Powell Conjugate Gradient	97.10%	0:05
Bayesian Regularization	98.30%	10:00
Quasi-Newton	96.90%	0:53
Resilient Backpropagation	97.50%	2:04
19 Levenberg-Marquardt Distributed Neural Networks	99.8%	0:30

Figure 6: Comparison of Training Methods

As discussed in the previous section, after creating a tool which has near 100% branch detection accuracy, a

visualization tool had to be created for the operators to see real time updates. The testing input file was used to do at home testing to see the accuracy of the tool. Once, it was doing well with this file, the next step was to take it into the lab for proper testing. The matlab was connected the server of the RTDS. This was done using HTTP two way connection with the server of the RTDS server. This RTDS system within the power lab, emulates a real time power grid and offers real time simulation data for any research project. So, the group visualization was connected to this program for testing. As seen below in figure 7, the RTDS interface looks very similar to the power grid images which were provided as the grid for the project. At the bottom of the screen, small rectangles can be observed, these are the on and off switches which the user can toggle between to change the status of the transmission lines real time. First, one line was turned off on the Area 1 side and the visualization instantly turned red for the corresponding line. This line was turned back on and it turned back to green instantly. Then a line from area 2 was turned off and back on and the visualization showed the correct corresponding The true test of the operation of the visualization was turning off two lines at the same time to see how well operation stands up under this situation. The visualization performed up to expectation and immediately turned red on the two lines which were turned off in the RTDS simulation tool and then back to green when turned on. As seen in figure 8, there are two red lines in the visualization tool, this figure was taken from the simulation time in the lab. Therefore the group has determined that the visualization works under all real time circumstances which the RTDS simulation software can produce for the power system.

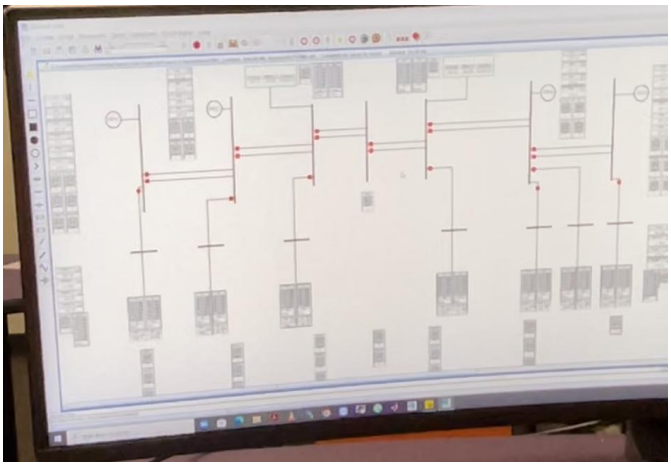

Figure 7: RTDS simulation tool

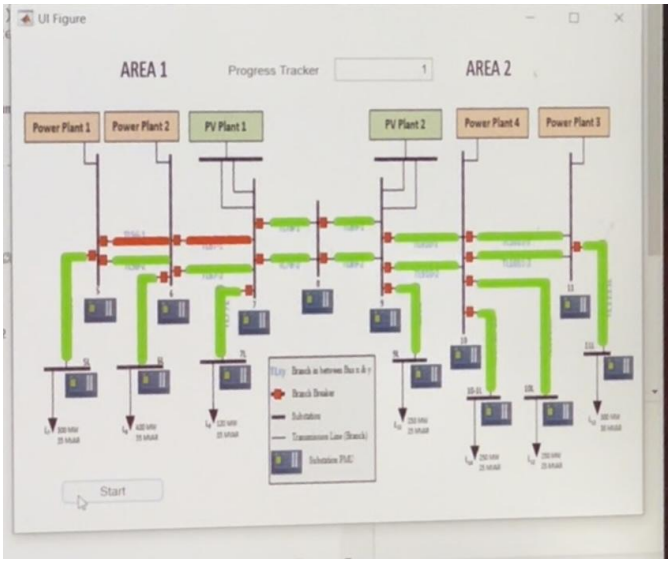

Figure 8: Visualization tool with 2 lines off

V. Conclusion

In todays transition to a world with a smart power grid, real time data analysis needs to be completed by power operators to determine when there are events on the grid. These could be big or small events, both of which could do large harm to the power grid. This can help to prevent blackouts such as what the United States saw during August 2003. It allows for the grid to accomplish some of the goals outlined by congress in the EISA bill of 2007. These include being fault tolerant, self healing, more reliable and using all sources of generation. Therefore all of the steps accomplished within the project help to show hypothetically how a power grid operator can address and show event detection.

Using the Matlab neural network toolbox, we were able to create a learning environment which could analyze the input and output data for both the training and collected data sets. It allowed us to get close to 100% accuracy in detecting the events in different branches of the power grid. This is a very high accuracy, which is what is needed for a practical usage in the real world grid. We achieved this using a feedforward neural network which was explained in detail above from the literature review and methodology explanation. It uses the Levenberg-Marquardt algorithm to solve for the non linear least squares equation to fit the training data provided for knowledge gain of the neural network. We achieved this goal of getting it near perfect by creating 19 separate neural networks, one for each sensor along the transmission branch diagram. This allowed for a high percentage of accuracy in detecting these events, by creating a distributed architecture. To improve the accuracy even further, different algorithms can be tested with the 19 neural network structure to see if there can be an increase in accuracy. Also, integrating the voltage magnitude data can help increase the accuracy of the system and help achieve the last 0.2% of accuracy needed to make a perfectly accurate system.

Having a visualization tool which is properly functioning is also a goal which the group achieved. So, along with the

accuracy of the neural network looking at the real time data, it was able to be given an output to the user in real time for if the different transmission lines were in proper operation or not. This was done by using a simple red and green coloring over the grid image, making it very easy for the operator to spot when problems arise. Therefore the group feel as though all set after goals have been accomplished and demonstrated thoroughly. This project was a great example of how to create a real time branch event detection system which can be implemented across a larger grid in real life.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Dr. G. Kumar Venayagamoorthy, instructor of the ECE 6160 course, Smart Grid for assisting with the material needed for this semester project. The authors would also like to thank TA Dulip Madurasinghe, for assisting with ideas and feedback throughout the project process and helping to run the laboratory equipment when simulations were needed. Lastly, we would like to thank the Clemson University Holcombe Department of Electrical and Computer Engineering for allowing students to use such world class simulation equipment to demonstrate class concepts within their power laboratory.

REFERENCES

[1] D. Madurasinghe, P. Arunagirinathan and G. K. Venayagamoorthy, "Distributed Identification of Power System Network Branch Events," *2020 IEEE Power & Energy Society General Meeting (PESGM)*, 2020, pp. 1-5, doi: 10.1109/PESGM41954.2020.9281775.

[2] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," in IEEE Potentials, vol. 13, no. 4, pp. 27-31, Oct.-Nov. 1994, doi: 10.1109/45.329294.

[3] G. Lera and M. Pinzolas, "Neighborhood based Levenberg-Marquardt algorithm for neural network training," in IEEE Transactions on Neural Networks, vol. 13, no. 5, pp. 1200-1203, Sept. 2002, doi: 10.1109/TNN.2002.1031951.

[4] "What is a neural network?," *What Is a Neural Network? - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/discovery/neural-network.html. [Accessed: 17-Mar-2022].