Detection of False Data Injection Attacks In Smart Grids

Kancharlapalli Lokesh[1], Harichandan.R[2], Rajesh.M[3], Lakshmipriya.S[4]
[1, 2, 3] Dept of Computer Science & Engineering
[4]Assistant Professor, Dept of Computer Science & Engineering
[1, 2, 3, 4] KCG College of Technology Chennai, India.

Abstract- *Due to the integration of advanced signal processing, communication, and control technologies, smart grid relies on a critical cyber infrastructure that is subject to adversarial cyber threats. In the existing system, KALMAN filter is used to detect the variations in the cumulative sum of the particular consumer premises and variation in voltage – current features is also being tabulated. The pre-stored data is hacked and rearranged by the unknown user. These kinds of attacks can be detected in the smart grid using KALMAN Filter. In the proposed system, design of Deep learning neural network algorithm is utilized to detect the malware present in the cumulative sum data through an iterative comparison of deep learning neuron models. Java based web application front-end is implemented to evaluate the user interface separately and admin interface separately.*

Keywords- Terms-kalman filter, random –forest classifier, baysian regulation algorithm, front-end, back-end

I. INTRODUCTION

Malware is short for "malicious software": hostile applications that are created with the express intent to damage or disable mobile devices, computers or network servers. Malware's objectives can include disrupting computing or communication operations, stealing sensitive data, accessing private networks, or hijacking systems to exploit their resources. The exponential growth in email and internet use over the last decade has brought with it a corresponding growth in malware.

Malware is deliberately malevolent, even when disguised as genuine software from a seemingly reputable source. Today's malware primarily targets sensitive personal, financial or business information, typically for monetary gain. Other objectives include identity theft and espionage, or service disruption targeting specific companies. The victims can just as easily be governments, enterprises or individual users. For a malware program to accomplish its goals, it must be able to run without being detected, shut down or deleted.

II. LITERATURE SURVEY

Chris Simmons proposed AVOIDIT: A Cyber Attack Taxonomy. Cyber- attacks have greatly increased over the years, where the attackers have progressively improved in devising attacks towards a specific target. To aid in identifying and defending against cyber- attacks we propose a cyber-attack taxonomy called AVOIDIT (Attack Vector, Operational Impact, Defense, Information Impact, and Target). We use five major classifiers to characterize the nature of an attack, which are classification by attack vector, classification by attack target, classification by operational impact, classification by informational impact, and classification by defense. Our fifth category, classification by defense, is used to provide the network administrator with information of how to mitigate or remediate an attack. Contrary to the existing taxonomies, our taxonomy efficiently classifies blended attacks. Our taxonomy is applied using an application approach with pabulum to educate the defender on possible cyber-attacks.

David Klaper[1] proposed A Taxonomy and a Knowledge Portal for cyber-security. Smart government is possible only if the security of sensitive data can be assured. The more knowledgeable government officials and citizens are about cyber-security, the better are the chances that government data is not compromised or abused. In this paper, we present two systems under development that aim at improving cyber-security education. First, we are creating a taxonomy of cyber-security topics that provides links to relevant educational or research material. Second, we are building a portal that serves as plat- form for users to discuss the security of websites. These sources can be linked together. This helps to strengthen the knowledge of government officials and citizens with regard to cyber-security issues. These issues are a central concern for open government initiatives.

Matthew Peacock [9] proposed Security Issues with BAC net Value Handling. Building automation systems, or building management systems, control services such as heating, air-conditioning and security access in facilities. A

common protocol used to transmit data regarding the status of components is BAC-net. Unfortunately, whilst security is included in the BAC-net standard, it is rarely implemented by vendors of building automation systems. This lack of attention to security can lead to vulnerabilities in the protocol being exploited with the result that the systems and the buildings they control can be compromised. This paper describes a proof-of-concept protocol attack on a BAC-net system and examines the potential of modeling the basis of the attack.

III. TYPES OF MALWARES

Malware is the collective name for a number of malicious software variants, including viruses, ransom ware and spyware. Shorthand for malicious software, malware typically consists of code developed by cyber attackers, designed to cause extensive damage to data and systems or to gain unauthorized access to a network. Malware is typically delivered in the form of a link or file over email and requires the user to click on the link or open the file to execute the malware. Malware has actually been a threat to individuals and organizations since the early 1970s when the Creeper virus first appeared. Since then, the world has been under attack from hundreds of thousands of different malware variants, all with the intent of causing the most disruption and damage as possible Malware delivers its payload in a number of different ways. From demanding a ransom to stealing sensitive personal data, cybercriminals are becoming more and more sophisticated in their methods. The following is a list of some of the more common malware types and definitions.

Types of Malware:

Virus

Possibly the most common type of malware, viruses attach their malicious code to clean code and wait for an unsuspecting user or an automated process to execute them. Like a biological virus, they can spread quickly and widely, causing damage to the core functionality of systems, corrupting files and locking users out of their computers. They are usually contained within an executable file.

Worms

Worms get their name from the way they infect systems. Starting from one infected machine, they weave their way through the network, connecting to consecutive machines in order to continue the spread of infection. This type of malware can infect entire networks of devices very quickly.

Spyware

Spyware, as its name suggests, is designed to spy on what a user is doing. Hiding in the background on a computer, this type of malware will collect information without the user knowing, such as credit card details, passwords and other sensitive information.

Trojans

Just like Greek soldiers hid in a giant horse to deliver their attack, this type of malware hides within or disguises itself as legitimate software. Acting discretely, it will breach security by creating backdoors that give other malware variants easy access.

Ransom ware

Also known as scareware, ransom ware comes with a heavy price. Able to lockdown networks and lock out users until a ransom is paid, ransom ware has targeted some of the biggest organizations in the world today — with expensive results.

IV. SYSTEM DESIGN

MATLAB: is a technical computing environment for high-performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing (via the Signal Processing Toolbox), and graphics into an easy-to-use environment where problems and solutions are expressed just as they are written mathematically, without much traditional programming. The name MATLAB stands for matrix laboratory. We will use MATLAB in ECE 360 in order to illustrate the concepts of digital signal processing with numerical examples. Each homework assignment will include some optional problems that require Mat-lab solutions. You will quickly realize that Mat-lab can often be used to check solutions to other problems as well. This is perfectly legitimate, but you still must turn in your analytical solutions for the pencil-and-paper problems MATLAB is available in DECS labs on a UNIX, PC, and MAC platform. You can invoke MATLAB by double-clicking on the MATLAB-Icon (MAC, PC) or by typing mat-lab on the Unix command line. You will then get access to the MATLAB command line, denoted by ">>".
This document is by no means a complete reference. There are tutorial and reference manuals for Mat-lab at the library.

The MATLAB System It comprises of five main parts:
The MATLAB language:
It is an array or matrix language at a higher level with control flow statements, functions, input/output, data structures, object-oriented programing features, etc. It permits both small

programming for creating fast and junk throw-away programs, and big programming for creating difficult and big application programs.

The MATLAB ambiance:

It has a set of tools offering lots of provisions that perform with as the MATLAB user or programmer. It provides help for variable management in your workspace and data transfer. Developing, managing, debugging, and profiling M-files can be done using the tools of MATLAB.

Graphics management:

Offering higher level commands for two dimensional and three-dimensional data visualization, animation, image processing, and presentation graphics are included in the Graphics management. Low-level commands for allowing full customization view of graphics for building Graphical User Interfaces on your MATLAB applications.

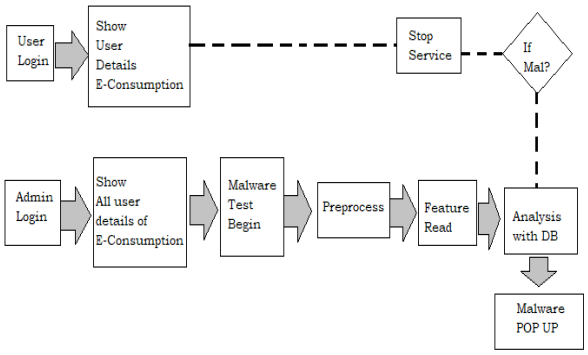
Mathematical library function of MATLAB:

There are various basic operations like sum, cosine, sine, and complex operations collection of computational algorithms for more sophisticated functions like matrix eigenvalues, inverse, fast Fourier transforms, and Bessel functions.

The MATLAB Application Program Interface (API):

C and Fortran programs for interacting with MATLAB is permitted by the library. There are other facilities included for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, MAT-files with reading and writing facility

V. DESIGN METHODOLOGY



VI. PROPOSED SYSTEM

In the proposed system, design of DOS attack detection through Machine learning algorithm is evaluated in which the malware in web applications is evaluated through a Test website

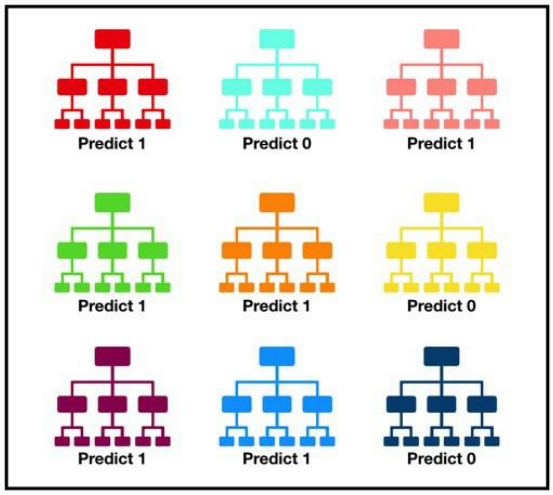
Design of Malware Data Visualization: This module is used to read the data, resize the data and visualize the raw data

Design of Random Forest Algorithm: This module is used to extract the data and analyze the input data with the database and display the malware injections present in it.

Design & Integration of Web App: This final module used to display the anamoly present in the web App and provide the immediate pop up when malware is detected also the design will perform DOS

The Random Forest Classifier

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).



Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple ***but powerful one — the wisdom of crowds. In data science*** speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

There needs to be some actual signal in our features so that models built using those features do better than random guessing.

The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

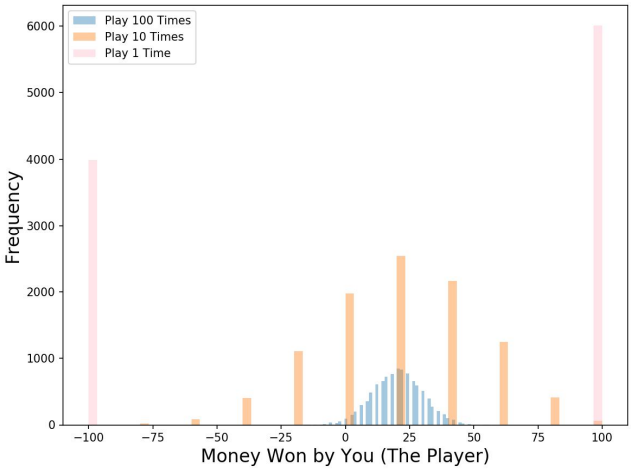
An Example of Why Uncorrelated Outcomes are So Great

The wonderful effects of having many uncorrelated models is such a critical concept that I want to show you an example to help it really sink in. Imagine that we are playing the following game: I use a uniformly distributed random number generator to produce a number. If the number I generate is greater than or equal to 40, you win (so you have a 60% chance of victory) and I pay you some money. If it is below 40, I win and you pay me the same amount. Now I offer you the the following choices. We can either:

1. **Game 1** — play 100 times, betting $1 each time.
2. **Game 2**— play 10 times, betting $10 each time.
3. **Game 3**— play one time, betting $100.

Which would you pick? The expected value of each game is the same:

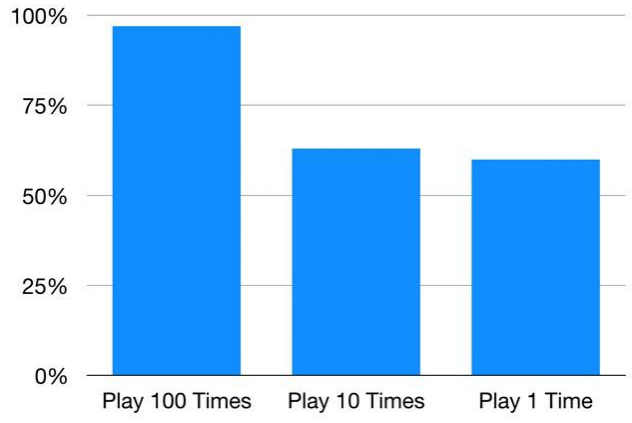
*Expected Value Game 1 = (0.60*1 + 0.40*-1)*100 = 20*
*Expected Value Game 2= (0.60*10 + 0.40*-10)*10 = 20*
*Expected Value Game 3= 0.60*100 + 0.40*-100 = 20*



Outcome Distribution of 10,000 Simulations for each Game

What about the distributions? Let's visualize the results with a Monte Carlo simulation (we will run 10,000 simulations of each game type; **for example, we will simulate 10,000 times the 100 plays of Game 1**). Take a look at the chart on the left — now which game would you pick? Even though the expected values are the same, **the outcome distributions are vastly different going from positive and narrow (blue) to binary (pink).**

Game 1 (where we play 100 times) offers up the best chance of making some money — **out of the 10,000 simulations that I ran, you make money in 97% of them!** For Game 2 (where we play 10 times) you make money in 63% of the simulations, a drastic decline (and a drastic increase in your probability of losing money). And Game 3 that we only play once, you make money in 60% of the simulations, as expected.



Probability of Making Money for Each Game

So even though the games share the same expected value, their outcome distributions are completely different. The more we split up our $100 bet into different plays, the more confident we can be that we will make money. As mentioned previously, this works because each play is independent of the other ones.

Random forest is the same — each tree is like one play in our game earlier. We just saw how our chances of making money increased the more times we played. Similarly, with a random forest model, our chances of making correct predictions increase with the number of uncorrelated trees in our model.

VI. RESULTS AND DISCUSSION

Write script-based tests to check that the outputs of MATLAB scripts, functions, or classes are as you expect. For example, you can use the assert function to test for actual output values that match expected values. Or you can test that the output variables have the correct size and type. To run your test scripts use the run tests function.

Write function-based tests to check that the outputs of MATLAB scripts, functions, or classes are as you expect. You can use a full library of qualification functions to produce four different types of test failures. For example, you can produce verification or fatal assertion test failures. Function-based tests subscribe to the xUnit testing philosophy

You can use the MATLAB performance testing framework to measure the performance of your MATLAB code. The framework includes performance measurement-oriented features such as running your code several times to warm it up and accounting for noise in the measurements.

The performance test interface leverages the script, function, and class-based unit testing interfaces. Therefore, you can perform qualifications within your performance tests to ensure correct functional behavior while measuring code performance. Also, you can run your performance tests as standard regression tests to ensure that code changes do not break performance tests.

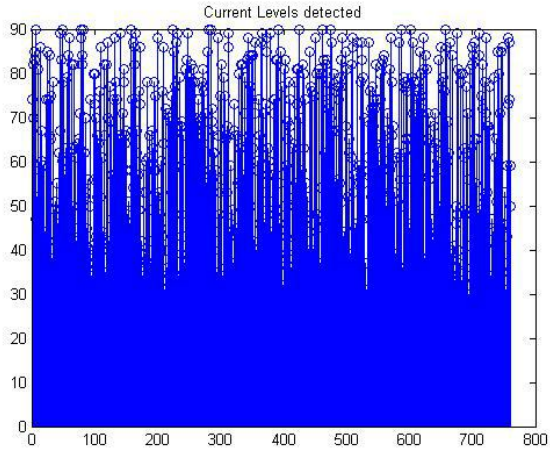
Measured estimate of the framework overhead

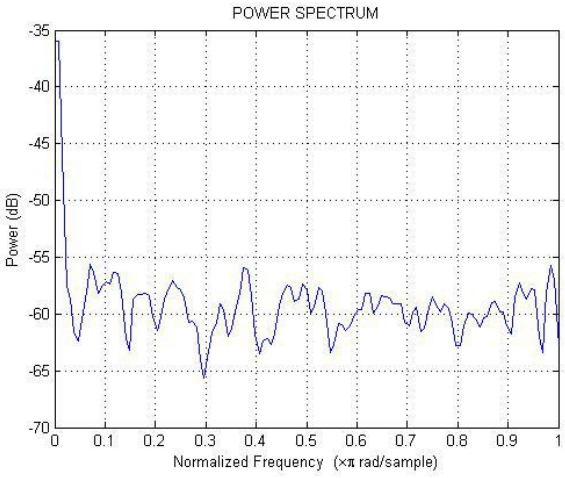
Class-based deriving from matlab. perftest.Test Case and using start Measuring and stop Measuring methods

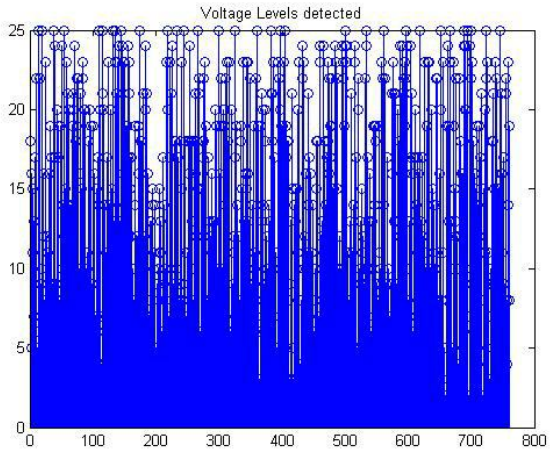
Code between calls to start Measuring and stop Measuring in each method tagged with the Test attribute Code outside of the start Measuring/stop Measuring boundary

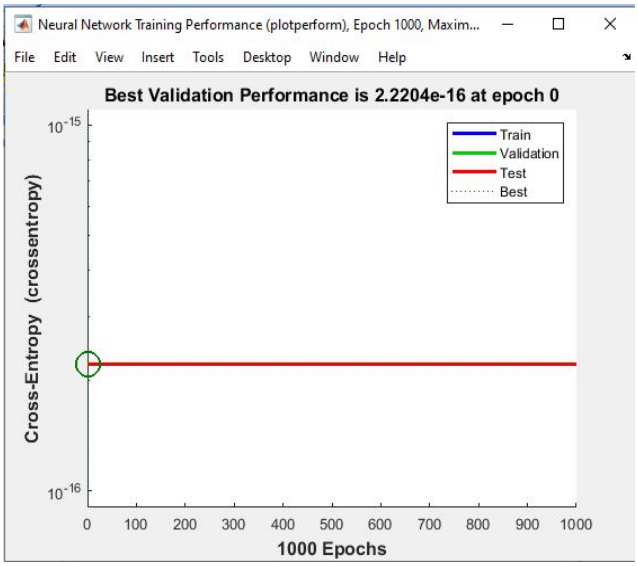
VII. CONCLUSION

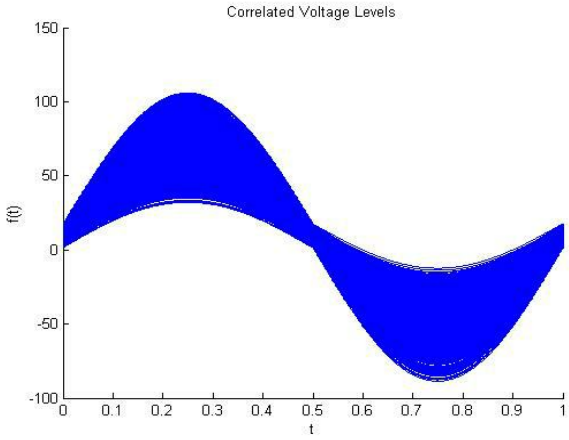
In this paper,a framework for the smart grid system using the kalman filter estimator together while using baysian regulation algorithm Here we detecting the attack what kind of attack will happened we using the Attack called randsome were attack it could be found by random forest classifier. from the survey of the base, supporting and additional papers key changes to the implementation method has been noted.the papers haved provided information needed to narrow down the implementation methods and provided the factors that can affect the parameters, literature survey provides a comparision of the vital factors to ensure the success of the developed software

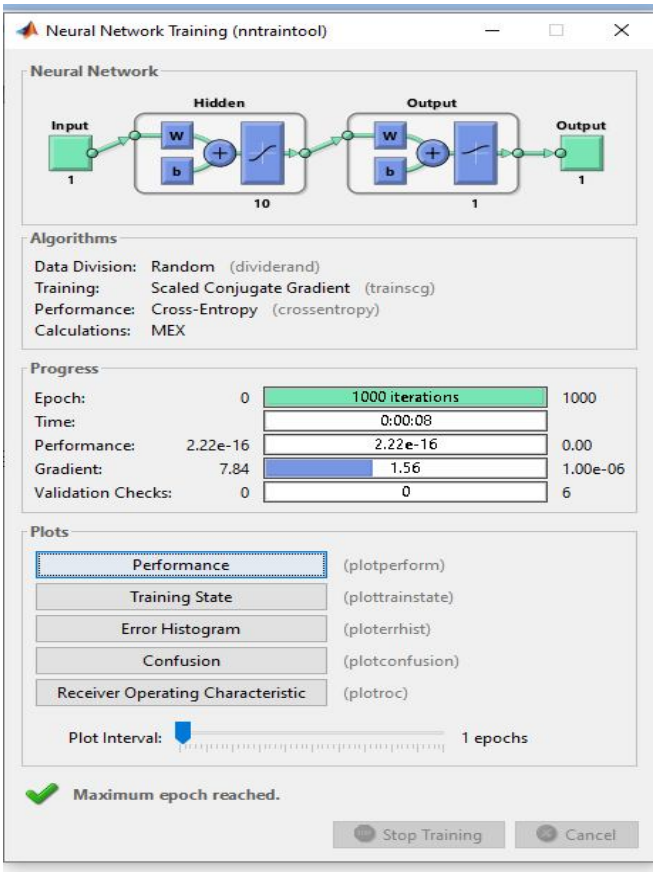


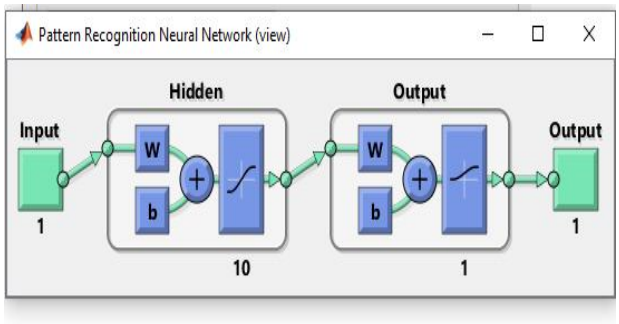


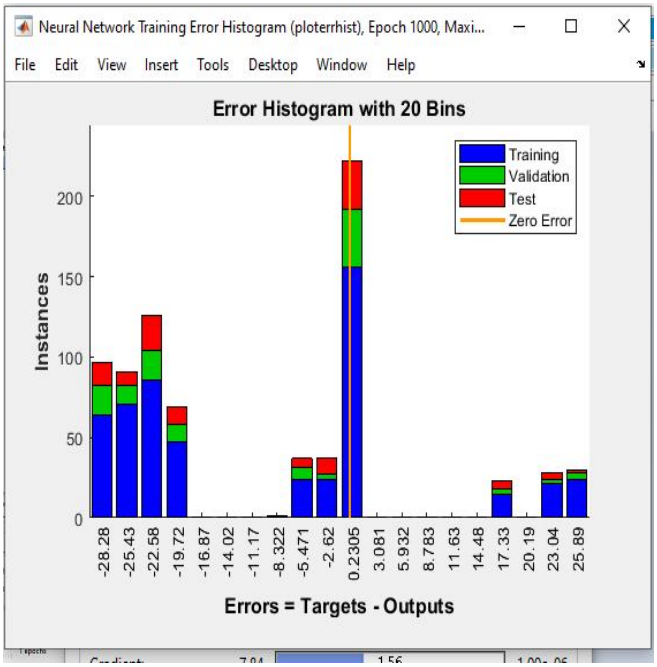

Voltage Levels detected
25
20
15
10
5
0
0
100
200
300
400
500
600
700
800

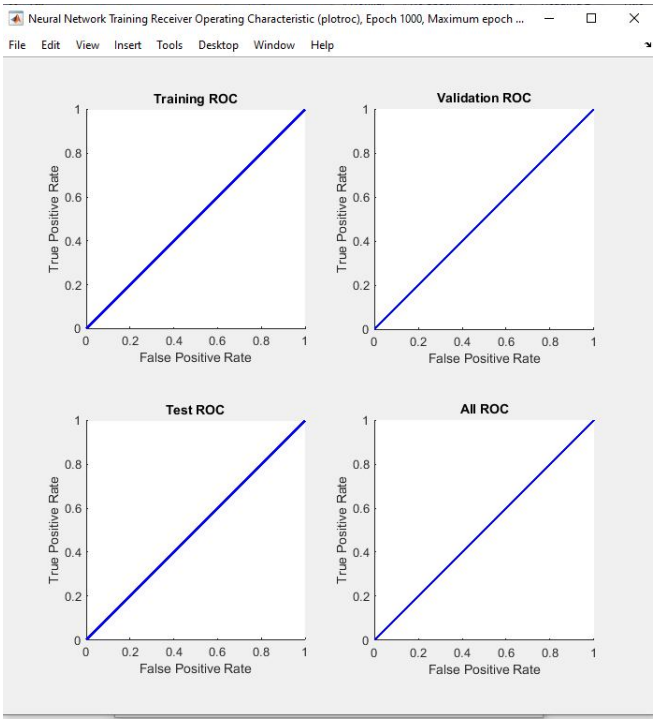

Neural Network Training Performance (plotperform), Epoch 1000, Maxim...
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 2.2204e-16 at epoch 0
Train
Validation
Test
Best
Cross-Entropy (crossentropy)
10^-15
10^-16
0 100 200 300 400 500 600 700 800 900 1000
1000 Epochs

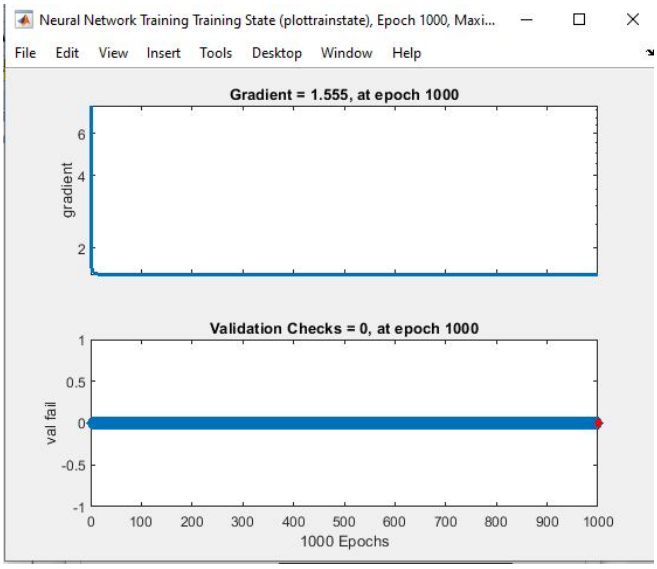

Correlated Voltage Levels
150
100
50
0
-50
-100
f(t)
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
t


Neural Network Training (nntraintool)
Neural Network
Input
Hidden
Output
Output
W
b
1
10
1
1
Algorithms
Data Division: Random (dividerand)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Cross-Entropy (crossentropy)
Calculations: MEX
Progress
Epoch: 0 1000 iterations 1000
Time: 0:00:08
Performance: 2.22e-16 2.22e-16 0.00
Gradient: 7.84 1.56 1.00e-06
Validation Checks: 0 0 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Confusion (plotconfusion)
Receiver Operating Characteristic (plotroc)
Plot Interval: 1 epochs
Maximum epoch reached.
Stop Training
Cancel


Pattern Recognition Neural Network (view)
Input
Hidden
Output
Output
W
b
1
10
1
1







REFERENCES

[1] X.Wang and P.Yi,"Security framework for wireless communication in smart distribution grid,'IEEE trans. Smart Grid ,Vol.2,no.4,pp.809-818,Dec.2011

[2] Y.Zhang,L.Wang,W.Sun,R.Green,and M.Alam,"Distributed intrusion detection system in a

multi-layer network architecture of smart grids,"IEEE Trans.Smart Grid,Vol.2,no.4,pp.796-808,Dec.2011

[3] B.Sikdar and J.Chow,"Defending synchrophasor data networks against traffic analysis attacks,"IEEE trans.Smart Grid,vol.2,no.4,pp.819-826,Dec.2011

[4] B.Brumback and M.Srinath "A chi-square test for fault – detection in kalman ilters,"IEEE Trans.Autom.Control,Vol.32,no.Ac-6,pp.552-554,Jun.1987

[5] Y.Moand B.Sinopoli,"False data injection attacks in control systems,"in Proc.1st Workshop Secure control Syst.,2010,pp.1-6,preprints

[6] C.Alcaraz,C.Fernandez-Gago and J.Lopez,"An early warning system based on reputation for energy control systems,"IEEE Trans .Smart Grid,Vol.2,no.4,pp.827-834,Dec.2011

[7] S.Bi and Y.J Zhang,"Defending mechanism against false-data injection attacks in the power state estimation ,"in Proc.IEEE GLOBECOM Workshops,Dec.2011,pp.1162-1167

.