Texas A&M University at Qatar

ECEN 403 - Electrical Design Lab 1

Semester: Fall 2019

Functional Modelling Report

Anomaly Detection in BACnet Protocol Systems

Team Members: Sofian Ghazali

Muhammed Zahid Kamil

Rahul Balamurugan

Project Mentors: Dr. Hussein Alnuweiri

Mr. Salah Hessien

Submission Date: 19/11/2019

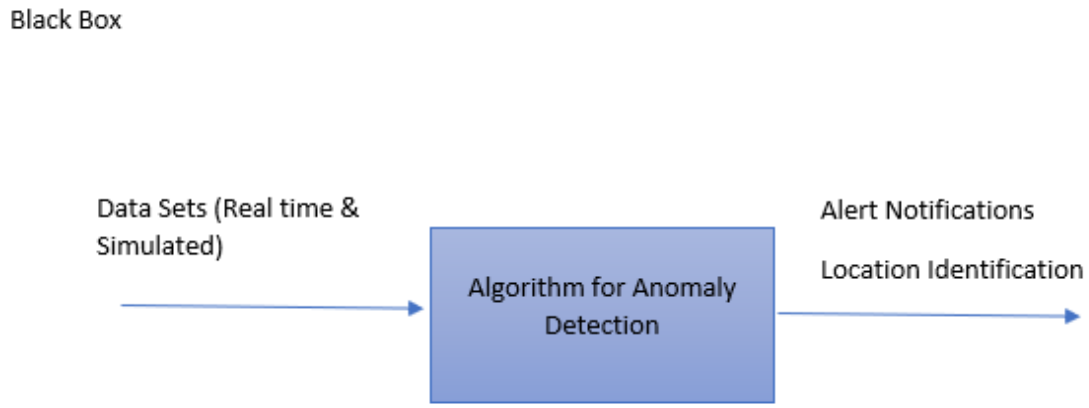
"An Aggie does not lie, cheat, or steal, or tolerate those who do."

TABLE OF CONTENTS

Introduction

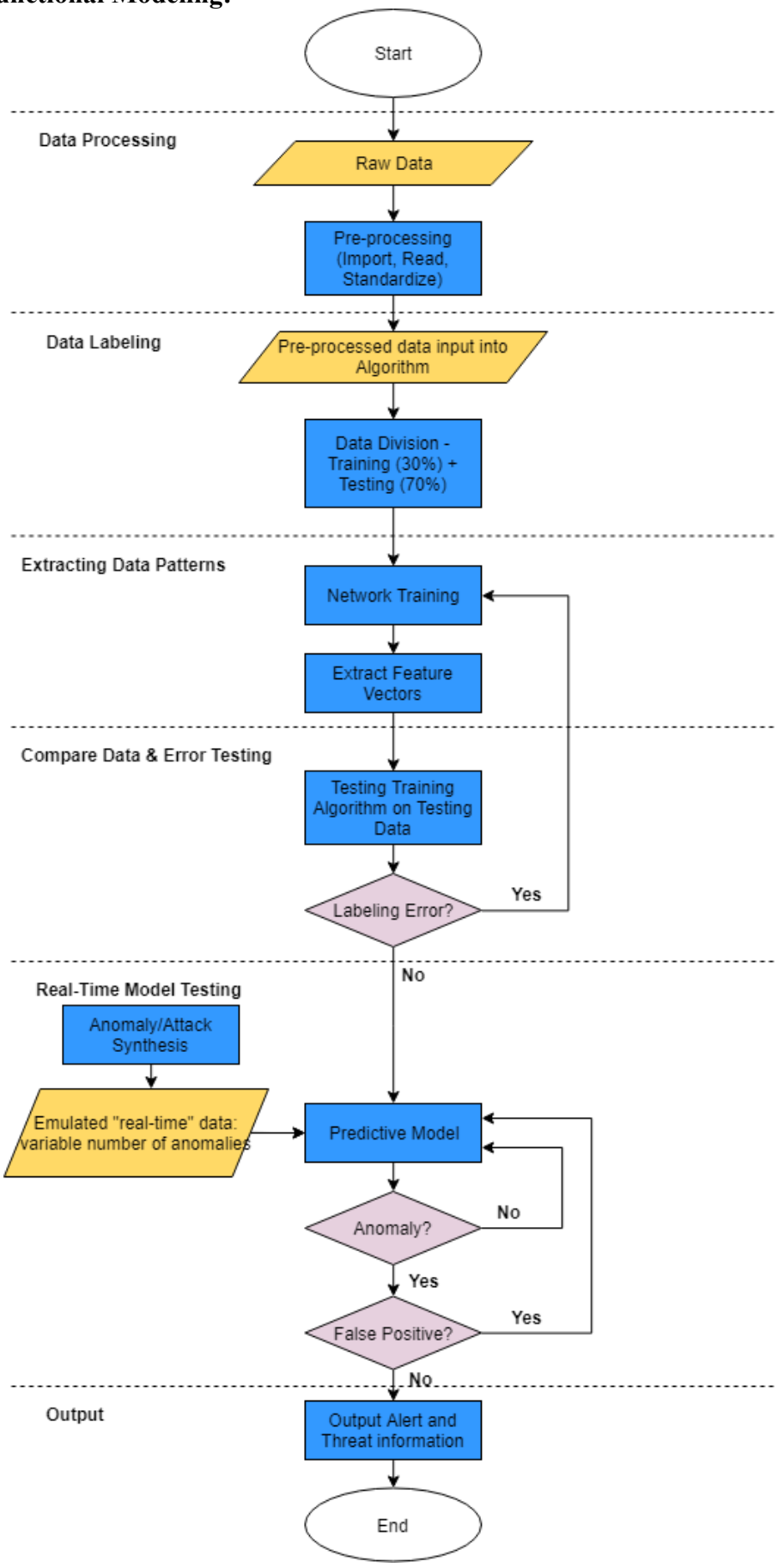
The objective of this assignment is to develop a functional structure of our project design. It is a great tool for us to convey our ideas through a visual depiction of the processes that go into designing our project. It allows us to elucidate the intricate details of our project and aid in team productivity. For a complicated project that involves huge data collection and running algorithms, it would've been difficult for us to comprehend the amount of components that needs to be implemented to make this algorithm run successfully. Using graphical data representations techniques like flow-charts and black-boxes, we can now use it as a reference point to review and understand the components throughout the duration of our senior design project. In addition, using flowcharts allowed us to consider troubleshooting issues that may occur during the project. This includes recognizing false positive results and how we can handle it by making decision boxes to determine flow of process. This flow leads to a possible solution and this assignment has allowed us to consider the possible solutions proactively, thereby allowing us to be productive in the future.

Upper Level Functional Model



Material & Energy Flows

Detailed Functional Modeling:



Below is the list of sub-functions obtained from the flowchart. We will further discuss about the function of each phase represented as sub-functions.

NOTE: We are planning to use Python to implement all the algorithms and methods discussed below.

I) Data Collection & Processing

This is the initial phase for our project. This step involves collecting the raw data from the relevant sources or higher authorities which is the first challenge we experienced. Usually for BACnet systems, there is a scarcity of data available due to building operators denying access to the BACnet traffic. This is mainly due to security purposes and so we had to think of ways to overcome this problem. Firstly, our mentor provided us BACnet traffic data that was obtained from QU. This data was labelled but was collected for a meager timeframe of 2 hours. This timeframe is insufficient to discern any variations in the dataset. Nevertheless, sufficient data was collected (~100,000) for us to kick-start our network training algorithm. This dataset volume might seem adequate but is not sufficient for our Deep Learning model. Usually, deep learning networks increase in performance as more data is fed in and models neural networks found in brain structures: learn by example [1].

An alternative method of data collection that we planned was to simulate our own BACnet dataset. This can be done by first observing normal patterns of traffic and then implementing Wireshark software. Wireshark collects simulated traffic data from Raspberry Pi microcontrollers that act as BACnet devices. In the future, we aim to approach a local company that specializes in BACnet related BMS systems and obtain real-time datasets. In case, data collection from the local company doesn't work, we will still be equipped with the simulated data from our Raspberry Pi stacks.

Once raw data is collected, we will pre-process or clean up the data for Deep Learning Networks to make sense of it. Pre-processing involves correcting raw data that is characteristized by inconsistencies, lack of certain trends and outliers. Through parsing techniques, we can resolve many of the above mentioned issues to make life easier for the Deep Learning Algorithm. To start off parsing, we first import the dataset into a programming software such as Python and categorize the dataset into a manageable format for the algorithm. Sometimes, we may be missing values and in this situation, we can choose to delete it provided that there is enough dataset. In some cases, we can calculate statistical values such as mean, mode or median t o replace those missing values since this can be a good approximation of the missing dataset.

II) Data Labelling

The generated data from the previous function block has to go through labeling for the program to be able to analyze it. We choose a semi supervised method to do this. First we would manually label ~30% of the first dataset, called the training data, using the THE classification model developed by Zheng et al. [2]. This model allows the traffic data to be classified into Time-driven, Human-driven and Event-driven data. The sets

of "normal" data organized at this step are passed to the anomaly detector so that it can learn to recognize the data patterns and extract the feature vectors from the data.

III) Extracting Data Patterns

This process is an instrumental step in the whole project because the algorithm trains on the given pre-processed and labeled dataset to recognize the regular data patterns and establish references for the detector to compare real time data against. This enhances performance, which can be represented in terms of efficiently labelling the un-labelled dataset from real-time and simulation.
The processes involved in this step include the labeling of the testing data in the Network Training sub-function; the extraction of feature vectors from the organized data in the second sub-function; and recursive learning to reduce the instances of erroneous labeling facilitated by error flagged data segments on a feedback loop from the next function block.

IV) Compare Data and Error Testing

Once the entire dataset (training + testing) is labeled, the processed data is then passed to this function block. The data in this step will be checked for errors by comparing with already labeled data from Zheng et al.'s project. The wrongly labeled instances thus identified would be flagged. The flagged data segments would then be run through the labeling algorithm recursively until the errors are eliminated. The training accrued over a few datasets should then enable the program to label further data with negligible error.

V) Real-Time Model Testing

The predictive algorithm refined by the preceding function blocks is tested by running "real-time" emulated data having both regular data and synthesized anomalies or attacks through the detection algorithm. As there are many kinds of anomalies, such as DoS attacks where an abnormality would show up in the volume of requests but not the requests themselves, we plan to implement an array of techniques to deal with many different anomalies. In general, however, the algorithm will compare the input "real-time" data's feature vectors with the ones identified previously as "normal" data patterns and alert the User if data segments show a significant variation from the norm. Those data segments are then read to identify the point of entry of the anomaly (sender's IP address) and the type of anomaly (if known). The information will be the output of the system.

VI) Output

The response we aim to get will be the location of the intrusion and type of attack. The location of the intrusion will be determined by our raspberry pi's since each raspberry pi device is a unique BACnet device and therefore will have unique BACnet IP address. We aim to include an LED in each BACnet device to show the location of the intrusion when the prototype is built. The type of attack to be determined will be compared with the existing attacks that we have trained our deep learning model with so that an immediate response will be taken.

Analysis and Discussion:

The model of the project as described in the previous section has helped us identify all the steps we need to take to be able to reach our proposed objective of creating a novel anomaly detector for BACnet traffic. To make our detector is truly novel, we realized that three features, found in other products separately, need to be available together in our system - high adaptability to new and emerging threats, high accuracy of anomaly detection, and a method to reduce or eliminate false positives. All of the three features mentioned are possible these days by selecting appropriate Machine Learning algorithms. We aim to select the algorithms and implementations for mainly the following:

1. Data pattern extraction: One of the hallmark features of any Machine Learning algorithm is to learn the features of the dataset and find a link between disparate features, thereby making valuable conclusions about the dataset. This is similar to how the human brain can connect experiences together to make cogent decisions when faced with a new situation.

2. Deep learning: This concept is mainly inspired from complex neural networks found in brain structures. Every time a novel information is observed by the brain, new synapse or link is made between brain cells, thus connecting different components during the learning process. This is the same method that deep learning networks do as they learn to form inferences and discern patterns from raw, unintelligible data. The more data we feed into the system, the better the performance.

3. integrating supervised and unsupervised methods: A technique that we aim to implement is the semi-supervised Machine Learning that involves the network algorithm to be trained initially with supervised learning methods, which involves learning from labelled dataset. After training has been completed, a new set of unlabeled data is fed and the algorithm is made to discern patterns and produce a labelling accordingly. If it produces error labelling, then this data is fed back into the algorithm to learn more about the dataset. This process continues until we achieve a suitable degree of accuracy, close to 95% (rough estimate).

4. anomaly detection: This is our project goal; we will now use our trained and tested algorithm to detect anomalies and flag data breaches. There is an important issue of false-positive results and this can impact the reliability of our algorithm. For this process, we plan to measure the false-positive rate by using a confusion matrix. This technique is composed of a table that describes the performance of the classification model and gives a visual representation of the performance of the algorithm. This technique will be valuable in making us aware of the degree of false-positive-ness and fine-tune our program as required.

As the topic of machine learning is very highly discussed on the internet, we found many frameworks out there that could possibly satisfy our design needs. These frameworks are available open source in GitHub repositories from which one can fork out their own implementations or use the available codes as in their projects. The chosen programs would then have to be adapted to our needs, which would additionally require us having a decent grasp of the programming language and the techniques involved. We are planning to either choose

implementations written in Python or ones that are compatible with Python. A few sources for all the information we may require to build our system are given below:

1. Coursera.org: This website hosts a variety of courses on different topics. The course we need from the lot, 'Machine Learning', is offered by Stanford, and is a complete introduction to all things machine learning related [3].
2. Github.com: This open-source website is a repository for a multitude of projects that covers nearly every topic related to computer science. For our needs, we would mainly require three applications- Tensor flow [4], which is an open source machine learning framework; and Scikit-Learn [5], which is an efficient tool for data mining and analysis built completely in Python. Lastly, we would be looking at Nervana- Neon [6], a very fast framework offered by Intel for deep learning. [7],[8],[9]

Conclusion

So far in this project, we have carried extensive literature review on the various Machine Learning Algorithms and found some good techniques that we will implement in our project. This assignment was a good starting point for us to consider the solutions and potential issues that we might face in the future. Flowchart has become a great tool for us to review our ideas that should go into a project and given us a clear direction. We will use our flowchart model to guide us through our project in the future. The explanation of sub-functions involved in our flowchart helped us review the technical details of our design and will help us in carrying out the design in an efficient manner possible.

References

[1] "What Is Deep Learning?: How It Works, Techniques & Applications," *How It Works, Techniques & Applications - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accessed: 17-Nov-2019].

[2] Z. Zheng and A. Reddy, "Safeguarding Building Automation Networks: THE-Driven Anomaly Detector Based on Traffic Analysis," 2017. [Online]. Available: http://cesg.tamu.edu/wp-content/uploads/2018/01/ICCCN-Zhiyuan-Zheng-Paper-2017-July-1.pdf. [Accessed: 08- Sep- 2019].

[3] Andrew Ng (instructor). "Machine Learning," www.coursera.org. Available: https://www.coursera.org/learn/machine-learning?ranMID=40328&ranEAID=SAyYsTvLiGQ&ranSiteID=SAyYsTvLiGQ-UMudRPTt5q9e0yTBEZ19_w&siteID=SAyYsTvLiGQ-UMudRPTt5q9e0yTBEZ19_w&utm_content=10&utm_medium=partners&utm_source=linkshare&utm_campaign=SAyYsTvLiGQ. [Accessed 17-Nov-2019].

[4] Abadi et al. "TensorFlow: Large-scale Machine Learning on Heterogeneous Distributed Systems," 2015. White paper, Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf [Accessed 18-Nov-2019].

[5] Pedregosa et al., "Scikit-learn: Machine Learning in Python," JMLR 12, pp. 2825-2830, 2011. [Online]. Available: http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf. [Accessed 18-Nov-2019].

[6] Carey Kloss. "Nervana Engine Delivers Deep Learning at Ludicrous Speed!" 18-May-2016, blog, www.intel.ai. Available: https://www.intel.ai/nervana-engine-delivers-deep-learning-at-ludicrous-speed/#gs.gw1cwk. [Accessed 18-Nov-2019].

[7] Tensorflow/tensorflow. Github repository: https://github.com/tensorflow/tensorflow. [Accessed 18-Nov-2019].

[8] Scikit-learn/scikit-learn. Github repository: https://github.com/scikit-learn/scikit-learn [Accessed 18-Nov-2019].

[9] NervanaSystems/neon. Github repository: https://github.com/NervanaSystems/neon. [Accessed 18-Nov-2019].