Handling Big NetFlow Data and Classification

InChan, Hwang
Kennesaw State University
Institute of Data Science and Analytics
Marietta, GA, U.S.A
ihwang@students.kennesaw.edu

PoChun, Lu
Kennesaw State University
Computer Science
Marietta, GA, U.S.A
plu5@students.kennesaw.edu

***Abstract*— The application of Deep Neural Network into the classification of Netflow packets into an ordinary and malicious one is an interesting challenge. The automation of its classification with a neural network promises high accuracy and speed for a massive traffic monitoring where human effort cannot be effectively practiced. Compared with an existing CNN based classification, our proposed model is simpler but denotes equal or higher accuracy on the classifications of Netflow data.**

Keywords—Security, Neural network, Machine learning, classification, Deep learning

I. INTRODUCTION

Network Intrusion Detection systems (NIDs) is an automated device tool to protect the local area network(LAN) of computers and the terminal machines in the LAN from an anonymous attack from the Internet. Hackers employ a variety kinds of methods, such as Probe, sending an Phishing Email to the users to collect private information like an bank account info. They also utilize U2R as a common type of attack. Attackers log into a system as a normal user, but they exploit a system bug and acquire the root privilege, and damage the system. Some of them launch R2L attack. When an user receives and runs an back door malware over an disguised email, the backdoor allows the attacker to get into the system and acquire the root privilege, damaging the user's computer. Another common type of cyber-attack is Distributed Denial of Service(DDoS). When it occurs, hackers send a massive amount of connection request packets to a server and the local network around it. The attacked server is placed in the enormous flow of the connection request packets and its reply capacity is overflowed and eventually lose its network capability.

In order to detect network intrusions, the use of netflow packets draws attention among network intrusion detection specialists nowadays because traditional deep packet inspection requires a lot of computing resources when network traffic is large and quick. It arises a critical issue when packet payloads are encrypted. Netflow packet analysis could provide a great alternative to the traditional deep packet inspection technique. This project involves netflow packet analysis together with statistical and machine learning methods to formulate an automated threat detection strategy.

As an automated threat detection strategy, Machine learning(ML) classification algorithms have been applied in the threat detection strategy to tackle the problems associated with the traditional NIDs, such as monitoring a massive amount of the Internet packets, identifying variant forms of malware, independent from inaccurate human-heuristics. Support Vector Machine(SVM), Logistic Regression(LR), Random Forest(RF), and Decision Tree(DT) have already been put into practice and provided favorable accuracy.

In the era of Big data, data is consisted of thousands of variables. But for each classification scenario, only a fraction of the massive variables are needed and considered being the most relevant variables. This process to select meaningful variables is called "Feature Selection." There are some advantages of this process, such as better data visualization and understanding, smaller data storage and measurement. In this project, whether this technique helps improve the test accuracy in comparison with a well-organized data preprocessing.

L-2 Penalized Regression(Ridge Regression) -$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \frac{\lambda}{2n}\sum_{j=1}^{p}{\beta_j}^2$ penalizes β coefficients to near zeros producing better generality for better prediction. In other words, L2 penalty shrinks some variables near zero as λ becomes larger, producing simpler and more interpretable model.

An attractive advantage of Deep Neural Network(DNN) is that it learns data with correlations within. Usually, feature selection is an important issue in such classification algorithms whether it is KNN, Logistic Regression, SVM, DCNN. Those algorithms usually run with better performance with a less number of important inputs. It improves both the performance with less complexity of computation, and produces more accurate prediction. There have been the development of statistical algorithms in feature selection, such as Ridge Regression. Its specialized algorithms enable researchers and cyber security specialists to select the most effective individual variables on classification from the array of many factors. In this project, other data preprocessing steps were applied at the same time. The results were compared with the use cases of other ML techniques.

II. BACKGROUND

Netflow based intrusion detection usually well detects Dos/DDoS attack and virus attack. Monitoring netflow data allows the detection of malicious traffic in an early stage, which is advantageous over the Internet traffic based detection.

A. Dos/DDoS attack

DDoS attacks the target network with a massive amount of SYN packets which are connection requests. When the target responds SYN packets with the corresponding amount of ACK packets, the attacker does not reply ACK packets with the corresponding amount of SYN packets. Then, the target network device's memory becomes fully occupied because the target network device waits sent ACKs answered with SYN packets. A well-known strategy of the DDoS detection is finding an abnormal traffic pattern, which is triggered by a threshold value set by a network specialist.

However, the machine learning based strategy detects DDoS traffic with learning its distinguished feature in the netflow traffic.

B. Malicious code attack

Malicious code spreads over the netflow packets is either packet based propagation and network connection based attack. As for the packet based propagation, the trace of malicious code can be found in the network packets. Such attack can be detected by the analysis of ASCII code in the packets. With the observation of packet communication frequency to a specific IP address alongside with the port scanning event, one can also ensure there is a malicious code in operation.

C. Deep neural network

Deep neural network has been used to classify images and other complicated data while finding latent variables and its relations. While there are some other machine learning techniques, deep neural networks are expected to show the most superior performance because it has been already showing very competitive classification accuracy for image classifications, such as handwriting detection, face detection, and medical image detection. Although this project will not utilize convolutional neural network(CNN), it will use a deep neural network to classify netflow packets whether it has malicious traffic or not.

D. L2 Regularization

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \frac{\lambda}{2n}\sum_{j=1}^{p}{\beta_j}^2$$

Machine learning algorithms always have some dangerous scenarios when the machine learning model is over-fitted. It is a state when test accuracy is too low compared to the train accuracy. It happens when the dataset has many variables and the train dataset is small so that the model is insufficiently generalized so that it could not predict the new incoming data accurately. L2 regularization relieves such a scenario while penalizing each variables coefficient. λ is the user defined value on the degree of coefficient penalty. Penalized coefficients will turn the overfitting model into a more generalized model, improving the test accuracy. In this project, the degree of its effectiveness for the classification accuracy are going to be examined.

E. Dropout

Deep neural network sometimes has a large number of layers to work on more complicated problems. However, as the size of the neural network increases, it will highly possibly fall into an overfitting issue, and it increases training time. It also requires a significant amount of data for a proper training. One way to avoid this issue is that turning off some randomly chosen neurons' activations for each mini-batch iteration. For next iteration after another, some other randomly chosen neurons are turned off while other neurons are being trained. It prevents the deep neural model from being over-fitted.

Because it has been already showing very competitive classification accuracy for image classifications, such as handwriting detection, face detection, and medical image detection. Although this project will not utilize convolutional neural network(CNN), it will use a deep neural network to classify netflow packets whether it has malicious traffic or not

III. EXPERIMENTAL SETUP

In this section, we are going to explain the details of the dataset which is used in this paper, and the experiment environment setup to run the data analysis. Moreover, we will provide the details of the neural network model, such as deep learning library and its functions used.

A. NetFlow dataset[1]

In this paper, we did not collect the NetFlow dataset by ourselves, but we utilized the dataset that had already been presented in VAST 2013 challenge committee, which can be download from VAST 2013 website. They provide two different datasets, Netflow data and Network health and status data.

i. Netflow data

This dataset is the main one used to implement the analysis. This dataset contains approximately 70 million numbers of the NetFlow data, with 18 network features for each row. The network features are IP address, port number, timestamp, etc.

Because of the hardware limitation, we only pick out the first 300,000 data in this paper.

ii.Network health and status data

Accordingly to the document that provided by VAST 2013 committee, this dataset was generated by a health monitoring program. As the result, this dataset shows the status of the network flow into four categories: 1 (Good), 2 (Warning), 3 (problem), 4 (unknown).

After combining the Network flow data with the status in the second dataset, the complete dataset with 18 features, and the status of each row has been obtained.

B. Environment

Because the local machine does not have enough power to handle the big data and run the analysis, to speed up the process, "Google Colaboratory" was used as the development computing environment. The first reason for the selection is that this one is free. Google provides this free online computing resource for everyone who has a google account. One only needs to make a Google colab file with python code in a personal google drive, and almost every libraries and packages are supported for use in the user code. The second reason is it is powerful computing power. As I mention above, this environment almost contains every library, so One does not need to install packages in the local computer, which might not have enough computing resource. Beside, Google Colaboratory also provides GPU hardware for Tensorflow, which speeds up the training step.

C. Library/Package

In this paper, Tensorflow and python language are going to be used. Using pandas is going to be used mainly to handle the Netflow data after being read into Dataframe data structure. For a neural network analysis, tensorflow package is going to be used. With these libraries and

packages, we finish data preprocessing and build the model.

iii. Data preprocessing

In the Netflow dataset, not all of the feature values are numeric, but some are categorical, and some are other types, which cannot be used to train the model directly Therefore, these features are preprocessed into appropriate formats.

Tensorflow provides a library called “feature_column”. It provide several types of feature transfer method, such as “ Bucketized”, “Embadding”, and “Hash feature”. The details will be presented in the next section IV.

iv. Model

To build a model for the netflow data analysis, Keras library will be utilized, which is a part of tensorflow. It provides plenty of in-built functions for users to build a neural network model easily. For instance, to build a sequential model, “Sequential” from “Keras.model” needs to be imported.

IV. PROCESS & MODEL

In this section, we will provide details about the whole experiment process and about the model. The experiment results will be depicted in the next section.

A. *Experiment Process*

To get the whole complete Netflow data, first step, we pick out “firstSeenSrcIP” column in Netflow dataset, which indicates the source of the IP of the first package. To match the column “Receivedfrom” in the Network health and status dataset, and then we can get the status at the “ statusVal” in second dataset.

After status checking, in the second step, we have to pre-processing the data in order to convert all data into a format accepted by the model. In the Netflow dataset, “ ipLayerProtocolCode” column indicates the IP protocol type in a textural format. Using “embedding_column” function, which is provided by tensorflow, to embedding the categorical data into a tensorflow categorical format.

In the same way, we also do this operation on column “ firstSeenSrcIP” and “ firstSeenDestIP”, which are presented in <aaa.bbb.ccc.ddd> normal IP presenting type, in a textual format.

To sum it up, after removing one unused column “ parsedDate”, we now have 14 normal numerical features, 3 embedding features, and one status column as the input for the NetFlow analysis model.

B. *Model*

To use Keras built-in library to build Deep Neural Network, sequential model is imported first, and then add layers one wants.

First layer is the input layer, which reads the whole features data that we discussed in the previous section.

Second and third layer are dense layers, which all feature nodes are fully connected between the different layers. The active function we choose is “relu” function. “Relu” function only produce one output when the value is greater than zero. If the output value lower than zero, it will return zero. Moreover, we also add up L2 regularization in these layer to avoid an overfitting issue. The number of the hidden units decreases as data move into the next layer. The 2nd layer has 64 units, and the 3rd layer has 32 units, the 4th layer has 16 units. The output layer has 5 units. We pass through 16 dimension for the third layer, we pass through 10 dimension as input.

For the output layer, it is still a dense layer, but the active function we change to “softmax”, which classifies outputs based on a probability distribution.

Because the data has more than 2 classes, and does not start from index 0,which is not continuous, so we have to using “sparse_categorical_crossentropy”, not any binary loss function or normal categorical function. About the learning optimizer, using Adam with small learning rate 0.0001.

V. RESULT

A. *Heap map*

In the paper[2], the author mentioned that because of the large scale of the network features, they use “Feature Selection” method for higher training accuracy, which picks out the highest relevant feature. About our dataset, we included all features in the dataset. We showed its legends in the heat map in figure 1.

Although we used all the features that had been provide by the VAST, we still plotted the heat map to show the distribution of the dataset. The color heat map shows the correlations between features. In this heat map we can see that those features, with darker color, at the right down corner show stronger correlations

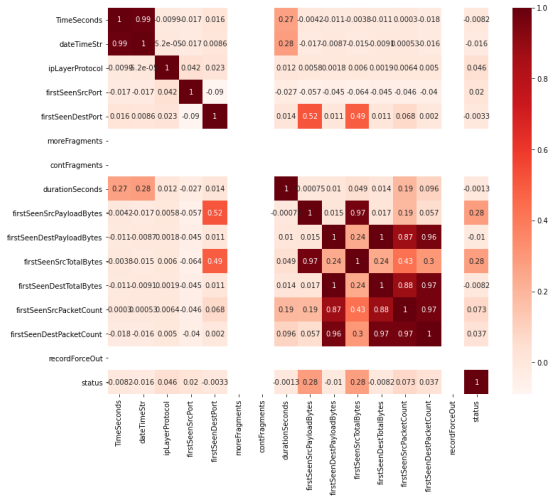
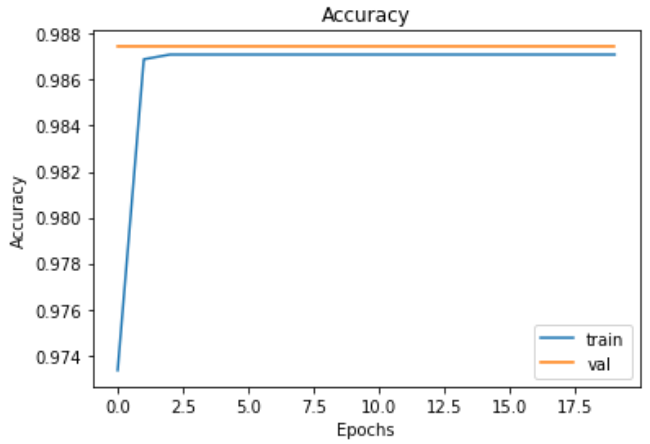


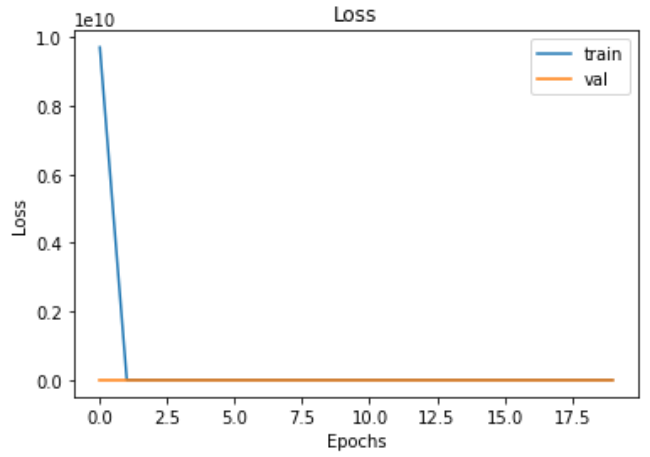
Figure 1. Shows correlations between features

B. *Result*

With the high performance platform “Google Colaboratory” and the embedding feature data from tensorflow, the average training time per epoch is about 9 second. The final results, after training 20 epochs, we get in this experiment is 98% accuracy in both training and testing datasets (figure 2). We can see that, for the training set, the loss and accuracy values showed high accuracy during the train. The convergence of the data happened in a really early stage, and finally can get the really high accuracy result.

There is a problem, we can see that during the training, the loss and the accuracy of the validation data set depict similar results. There are two reasons that may have caused this issue.






```
219/219 [==============================] - 0s 2ms/step - loss: 0.1127 - accuracy: 0.9859
Accuracy 0.9858571290969849
```

Figure 2. a), the accuracy of training data set and validation dataset. b) the loss of training dataset and validation dataset. c) the loss and the accuracy of the testing dataset.

1. *Overfitting*

The data may have an overfitting problem. That is probably the reason that the validation loss doesn't improve with the training dataset. The reason could be, too much features we had, need more regularization, or due to insufficient amount of data.

In this project, the model has already achieved over 95% of accuracy, so we don't think regularization could improve the accuracy further.

About regularization techniques, in the paper[3] shows that L2 Regularization is a better method than L1 one to perform regularization for a neural network. However, the result did not show too much difference after we add the method of L2 regularization.

2. *Data Unbalanced*

As the result, we think the most possible reason caused this problem is lack of data. In our dataset, the classes samples are really unbalanced: for class "1 (good)" there are 292,901 out of 300,000 number of dataset, which is the dominance class. The "2 (warning)" class only have 6,164 number of data. The "3 and 4" classed only had 971 instance. Due to the unbalance of the dataset, when splitting the data, it might give too much same classed instance to the validation dataset, even we have already called the shuffling function. KFold cross validation technique will be applied to the model in the future to clear out this issue.

VI. DISCUSSION

Overall, we conducted a partial experiment to analysis the Netflow. However, in some of our process there can be some discussions.

1. *The process of checking data status*

About the processing to check the status of each data, we just compared the IP address, but in the real world, there might be more information to consider the Netflow data's maliciousness.

2. *Data pre-processing method*

In this paper, the main pre-processing is embedding text features. We have used the tensorflow feature column method. However, we have limited knowledge about this process, which make our model inflexible. It has given us some difficult time to build the model.

VII. CONCLUSION

For the over the entire experiment, we think we have gained more knowledge about handling the Netflow data, data pre-processing, and building the Deep neural network. We obtained high accuracy from the model we built, which is about 97%. Because of the limitation of the knowledge in neural network, Netflow data, and statistics, we might have not chosen the best function and the methods. These are the direction for our future research.

VIII.REFERENCES

[1] Data Set [Online]. Available: http://vacommunity.org/VAST Challenge 2013: Mini-Challenge 3

[2] Liu, X., Tang, Z., & Yang, B. (2019, May). Predicting network attacks with CNN by constructing images from NetFlow data. In 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS) (pp. 61-66). IEEE.

[3] Overfitting and underfitting, Tensorflow tutorial[Online]. Available https://www.tensorflow.org/tutorials/keras/overfit_and_underfit?hl=ko.

[4] Nyasore, O. N., Zavarsky, P., Swar, B., Naiyeju, R., & Dabra, S. (2020, May). Deep packet inspection in industrial automation control system to mitigate attacks exploiting modbus/tcp vulnerabilities. In 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS) (pp. 241-245). IEEE.

[5] Dash, M., & Liu, H. (1997). Feature selection for classification. Intelligent data analysis, 1(1-4), 131-156.

[6] Tensorflow feature columns URL:https://www.tensorflow.org/tutorials/structured_data/feature_columns