Deep Learning Approach for Network Intrusion Detection

Computer Science B & Networking Essential Final Project

Professor: Abdallah Altrad

Chen Chanveasna
Cybersecurity Major
American University of Phnom Penh
Phnom Penh, Cambodia
2024023chen@aupp.edu.kh

***Abstract*—Network intrusion detection is very crucial for maintaining cybersecurity in our increasingly complex digital environments. This report shows a deep learning approach to network attack classification using state-of-the-art machine learning techniques. The study develops two neural network models trained on the TII-SSRC-23 and CIC-IDS2017 datasets, with a focus on comparing performance between imbalanced and balanced class scenarios. The feature selection was conducted using RandomForest feature importance extraction, which helps in identifying ten critical network traffic characteristics.**

The first model, trained on an imbalanced dataset, achieved an overall accuracy of 95%, with precision of 94% for benign traffic and 95% for malicious traffic. The second model, trained on a balanced dataset, demonstrated perfect classification metrics, achieving 99.95% precision, recall, and F1-score across both benign and malicious traffic categories. Key features for detection included network-related metrics such as packet length, flag counts, destination ports, and flow inter-arrival times.

Keywords—Network Intrusion Detection System, Deep Learning, scikit learn, Imbalance Model, Balanced Model, Malicious, Benign, Callback Function, DDoS, Nmap, Testing, Simulated Data.

I. Introduction

In today's digital era, everything is being conducted online be it studying, shopping, and selling. This results in digital networks technology growing and developing endlessly. However, like Isaac Newton's third law states "For every action, there is an equal and opposite reaction.", despite the benefits brought by the development of digital technology, it is also observed that more sophisticated types of cyberattacks also continue to rise, posing significant threats to individuals, business enterprises, organizations, and national security. The traditional network intrusion detection also face many challenges due to the inability to keep up with trend of cyberattacks.

This work aims to address the challenge of traditional network intrusion detection by leveraging deep learning techniques to classify network traffic using the TII-SSRC-23 and CIC-IDS2017 datasets for training. The features that are being selected for training is implemented through RandomForestClassifier from scikit learn. Additionally, this work also present an evaluation of the impact of balanced versus imbalance class distributions on deep learning model performance.

II. Related Works

Network intrusion detection has long been a concern and critical focus within the field of cybersecurity. And there are also many similar works done in this particular topic as well. For instance, in [1] research paper which critically discussed and analyzed the performance of traditional method and deep learning method on Network Intrusion Detection System. In addition, [2] is an existing work done on the topic of deep learning approach to Network Intrusion Detection which provided detailed insights for those who want to learn about the project. Last but not least, [3] Kaggle Notebook also presents a Machine Learning approach to Network Intrusion Detection System as well.

III. Methodology

A. Datasets

Two primary datasets were used in this research:

1) *TII-SSRC-23:* An extensive dataset of Malicious and benign traffic that is carefully compiled to aid the research and development of Intrusion Detection System (IDS) [4]. Csv file size is 5GB. For this work, only 25% and 10% of the dataset were used across two different scenarios.
2) *CIC-IDS-2018:* A dataset that was captured over a period of 5 days (Monday-Friday) with various type of malicious attack and benign traffic published by Canada Institute for Cybersecurity [5]. However, for this work, only Monday (pure benign traffic) was used in order to address the issue of class imbalance. Csv file size is 225MB.

B. Data Preprocessing

The preprocessing pipeline involves several critical steps such as:

1) *Data cleaning:* Replace both negative and positive infinite value in the CIC-IDS2017

Benign-Monday-WorkingHours dataset with NaN value and then drop all NaN values as well as fully duplicated values inside the dataset. Subsequently, dropping unwanted columns such as those with datat type object that contains identifier columns in TII-SSRC-23 except for the Label which will be mapped into binary classification format (0 for Benign, 1 for Malicious) after merging the two datasets together.

2) *Class Imbalance handling:* For the first model the malicious packets was downsampled to only 25% of the original malicious packets but class imbalance still occur. While the second model malicious packets was downsampled to only 10% of the original, which successfully achieve the objective of balancing the class.
3) *Data separation:* Separate the features and target (Label).
4) *Data scaling:* Utilizing the MinMaxScaler from scikit learn to scale all the numerical values.

C. Feature Selection

Fitting the separated data (all features, and Label) into the RandomForestClassifier model, and utilizing the feature importance function from scikit learn to extract only the top 10 most critical network features.

1) *The first model: As shown in Figure 1.1. The most critical network features for the first model:*
   - Bwd Segment Size Avg
   - Subflow Bwd Bytes
   - Bwd Init Win Bytes
   - Bwd Packet Length Max
   - Dst Port
   - Flow IAT Min
   - Bwd Header Length
   - RST Flag Count
   - Bwd Packet Length Mean
   - Fwd Packets/s

2) *The second model: As shown in Figure 1.2. The most critical network features for the second model:*
   - RST Flag Count
   - Dst Port
   - Bwd Packet Length Max
   - Bwd Packet Length Mean
   - Bwd Init Win Bytes
   - Flow IAT Min
   - Fwd Packet Length Min
   - Bwd Segment Size Avg
   - Subflow Bwd Packets
   - FWD Init Win Bytes

3) *Networking background: In order to get a better understanding, all the features should be briefly explained.*
   - **Bwd Header Length:** The average length of the header of the backward packets in a flow.
   - **Bwd Init Win Bytes:** The initial window size of the backward flow.
   - **Bwd Packet Length Max:** The maximum length of a backward packet in a flow.
   - **Bwd Packet Length Mean.** The average length of backward packets in a flow.
   - **Bwd Segment Size Avg:** The average size of segments in the backward flow.
   - **Dst Port:** The destination port number of the flow.
   - **Flow IAT Min:** The minimum inter-arrival time between packets in the flow.
   - **FWD Init Win Bytes:** The initial window size of the forward flow.
   - **Fwd Packet Length Min:** The minimum length of a forward packet in a flow.
   - **Fwd Packets/s:** The number of forward packets per second in the flow.
   - **RST Flag Count:** The number of packets with the RST flag set in the flow.
   - **Subflow Bwd Bytes:** The total number of bytes transferred in the backward direction of a subflow.
   - **Subflow Bwd Packets:** The total number of packets transferred in the backward direction of a subflow.

D. Model Building

A fully connected neural network was developed with the following key characteristics:

- Multiple dense layers with ReLU activation
- Batch normalization to improve training stability
- Dropout layers to prevent overfitting
- Sigmoid activation in the output layer for binary classification

E. Callbacks Function And Model Training

Before the training begins, it is crucial to split the dataset into three portions: one set for training, another for testing, and the last set for validation. The data splitting was performed using scikit learn's train_test_split function, with 20% of the data reserved for testing, another 20% for validation, and 60% for training.

1) *Callback implementation:* Two key callback functions: Early Stopping and Learning Rate Reduction were implemented in order to minimize overfitting and optimizing the training process.

   a) *Early stopping:* This callback monitors validation loss. It will automatically stop the training process if there is no improvement observed for n consecutive epochs. Where you can set the number of n.

b) *Learning rate reduction:* This callback dynamically reduces the learning rate by half whenever the validation loss plateaus. It allows for more fine-tuned learning.

2) *Training Process:* The model was trained with the following configuration:
   - Maximum Epochs: 100
   - Batch Size: 512
   - Callbacks: Early Stopping and Reduce LR
   - Verbose: 1 (to see the training process per epoch)

```
# Training the model
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    batch_size=512,
    epochs=100,
    callbacks=[early_stopping, reduce_lr],
    verbose=1  # Show epoch-by-epoch progress
)
```

F. Data Simulation For Testing

In order to observe the model performance in predicting real-world network traffic, a comprehensive testing approach was developed to simulate network attacks.

1) *Attack Method:* Using Hping3 to emulate DDoS Syn-Flood attack. Command: sudo hping3 -S –flood -p 80 [IP Address]. Use nmap scan to emulate network probe. Command: nmap [IP Address].
2) *Data Capture:* Capture the malicious and benign network traffic separately using Wireshark and save it as pcap file namely (benign.pcap, ddos_testing.pcap, nmap_probe.pcap).
3) *Data Preprocess*: Extract all the relevant features from pcap file using CICFlowMeter and preprocess it the same way as did for the training.

IV. Result And Performance Analysis

A. Imbalance Dataset Model

1) *Performance metrics on testing set:*
   - Overall Accuracy: 95%
   - Benign Traffic:
     - Precision: 94%
     - Recall: 78%
     - F1-Score: 85%
   - Malicious Traffic:
     - Precision: 95%
     - Recall: 99%
     - F1-Score: 97%

	precision	recall	f1-score	support
Benign	0.94	0.78	0.85	106053
Malicious	0.95	0.99	0.97	432870
accuracy			0.95	538923
macro avg	0.94	0.88	0.91	538923
weighted avg	0.95	0.95	0.95	538923

2) *Testing results on simulated data:*

	Accuracy		
Testing File	With Scaling	Without Scaling	Sample Size
benign.pcap	95.8490%	38.3018%	530
ddos_testing.pcap	0.1750%	99.9671%	9141
nmap_probe.pcap	90.4027%	99.9056%	3178

B. Balanced Dataset Model

1) *Performance metrics on testing set:*
   - Overall Accuracy: 99.95%
   - Benign Traffic:
     - Precision: 99.98%
     - Recall: 99.87%
     - F1-Score: 99.93%
   - Malicious Traffic:
     - Precision: 99.92%
     - Recall: 99.99%
     - F1-Score: 99.96%

	precision	recall	f1-score	support
Benign	0.9998	0.9987	0.9993	105520
Malicious	0.9992	0.9999	0.9996	173739
accuracy			0.9995	279259
macro avg	0.9995	0.9993	0.9994	279259
weighted avg	0.9995	0.9995	0.9995	279259

2) *Testing results on simulated data:*

	Accuracy		
Testing File	With Scaling	Without Scaling	Sample Size
benign.pcap	93.5849%	92.6415%	530

ddos_testing.pcap	0.0328%	91.3142%	9141
nmap_probe.pcap	1.7935%	1.2586%%	3178

V. Evaluation and Comparison

A. Performance

1) *Testing set:* Both models show excellent performance. However, the balanced model has higher overall accuracy, precision, recall, and f1-score compared to the imbalance model. Keep in mind that it is a testing set which means that there might be some patterns or any underlying factors that hugely play an important role in improving the model performance on the testing set. Let's observe how the model performs with simulated data.
2) *Simulated data:*
   - The imbalanced model: accurately predicted 95.84% correctly on benign.pcap, 0.175% on ddos_testing, and 90.4% on nmap_probe.pcap (with scaling applied). Accurately predicted 38.3% on benign.pcap, 99.96% on ddos_testing.pcap, and 99.90% on nmap_probe.pcap. This shows that the imbalance model could be predicted more accurately without scaling applied. The reason for that might be underlying factors such as outliers, and the overall packets flow values which influence the data in some way that hinders the performance of the model.
   - The balanced model: accurately predicted 93.58% correctly on benign.pcap, 0.03% on ddos_testing, and 1.79% on nmap_probe.pcap (with scaling applied). Accurately predicted 92.64% on benign.pcap, 91.31% on ddos_testing, and 1.25% on nmap_probe.pcap (without scaling applied). Although the balanced model performs very well in detecting benign.pcap, it fails to detect DDoS, and nmap attack when applied scaling. This is a challenge that machine learning or deep learning beginner face when building a project. The reason for that might have something to do with the method of data preparation before training (scaling, feature selections, outliers..etc).

Overall the performance of both models in predicting simulated is acceptable only when scaler is not applied. Additionally, the imbalance model might be a little biased toward the majority class. While the balanced model shows an incredible performance in identifying both malicious and benign simulated data. Although it fails to classify the Nmap probe as an attack it is still within an acceptable range as Nmap was never really a malicious network attack in the first place. Unless more flags are used like -T5 which is an aggressive scan tier offered by Nmap.

B. Applications

The two models presented in this project could potentially apply to a wide range of scenarios in cybersecurity, particularly in the controlled environment such as laboratories for studying and developing deep learning based Network Intrusion Detection System. Although some works need to be done in order to improve the overall performance of the models.

C. Complexity

The deep learning model is quite easy to train but trying to get the best performance out of it is quite hard. For example, in this project in order to improve the accuracy of the imbalanced model, more works have to be done such as downsampling it to 10% to address the class imbalance issue, and implementing another round of the feature selection process. It also requires a sufficient or basic knowledge of Networking/Cybersecurity concepts in order to build this project. Using the correct tools like hping3, nmap, Wireshark, and CICFlowMeter is very crucial.

D. Scalability

Both models could potentially be scaled to more large and robust models to process more extensive feature sets and identify more attack types instead of just malicious and benign. Although effective feature engineering, complex and advanced neural network architecture is needed.

E. Usability

The simple and straightforward architecture of the deep learning model, combined with clear preprocessing steps as well as a simple feature selection method enhanced the usability of this project. It can serve as a good example, demo, or working proof of concept for beginner cybersecurity/machine learning enthusiasts.

VI. Discussion

A. Challenge

As a beginner in machine learning/deep learning and cybersecurity, this project presented some challenges:

- Understanding the workflow or process of training a model
- Understanding complex neural network architecture
- Selecting appropriate features
- Risk of overfitting
- Lack of computational resources when working with huge datasets.

B. Lesson Learned

Through many trials and errors during the experimental process with machine learning, the project has finally shown some significant improvement when switched to deep learning, even though the accuracy of some scenarios still fell short. However, several key lessons were learned from this project which will serve as future avenues:

1) *The importance of feature selection:* The integration of machine learning (RandomForestClassifier) to conduct the feature selection proved to be essential in

improving the model performance as it narrowed down the input data to only the most relevant features. However, the process of training machine learning also has to be taken into deeper consideration.

2) *Importance of addressing class imbalance:* This project has effectively highlighted the significance of balancing the dataset. It also underscored the challenges of working with real-world data in cybersecurity. Even though oversampling or undersampling methods could be considered effective in addressing the class imbalance issue, they can also reveal potential biases in classification results.
3) *The sign of overfitting:* The balanced model, although demonstrated a high accuracy during the testing set when facing real-world (simulated) data the model performed very poorly. This indicates the overfitting issue of the model as it fails to generalize well to unseen data.

VII. Conclusion

This project could be considered to have successfully served as a working proof of concept that demonstrated the promising potential of deep learning techniques in network intrusion detection. It might not be perfect right now, but if given more time to experiment, its performance will be improving significantly. The comparative analysis of the imbalance and balance models has provided valuable insight into the significance of class balancing when preparing the dataset.

The simulation-based testing also highlights the importance of data preprocessing and feature selection. There might be outliers or hidden intricate characteristics, patterns, and relationships of different types of attack in the dataset that are overlooked due to the lack of exploratory data analysis, which is the sole reason that causes both models to perform poorly on the simulated data when applied scaling.

In conclusion, this project has provided valuable insights into the challenges and considerations involved in the development of deep learning approaches to network intrusion detection systems. It also shown that perfection is not achieved on the first attempt, but rather through numerous trials and errors.

VIII. Future Work

Upon gaining many valuable insights from this project, several promising ideas for future work can be explored:

- Enhancing Data Preprocessing: focusing more on the Exploratory Data Analysis (EDA) in order to effectively preprocess the data for training.
- Experiment With More Complex and Advanced Neural Network Architecture: investigate the potential of other deep learning models such as CNNs or RNNs.
- Expanding Scope of Attack Type: from binary classification to multiclass classification.
- Real-World Deployment: deploy the model in the real-world network environment.
- Continuous Evaluation: continuously updating the model to adapt to the evolving network traffic patterns and attack vectors.

IX. Acknowledgment

I want to thank my friend Mr. Sok Piseth, who is also working on the same project, for inspiring me to switch from a Machine Learning network intrusion detection system to a Deep Learning based instead. I found out that the deep learning model performs very well compared to machine learning even with the same methodology applied. I also would like to acknowledge the related or similar works done on this project that I mentioned in the Related Works section above. It is through their wonderful works that I am able to complete this project.

X. References

[1] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, June 2016, pp. 581-585.

[2] Moraboena, S., Ketepalli, G., & Ragam, P. (2020). A Deep Learning Approach to Network Intrusion Detection Using Deep Autoencoder. Rev. d'Intelligence Artif., 34(4), 457-463.

[3] https://www.kaggle.com/code/dhoogla/cic-ids2017-02-perfect-classification-binary

[4] D. Herzalla, W. T. Lunardi, and M. Andreoni, "TII-SSRC-23 Dataset: Typological Exploration of Diverse Traffic Patterns for Intrusion Detection," IEEE Access, vol. 11, pp. 33192-33213, 2023, doi: 10.1109/ACCESS.2023.3319213.

[5] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.