# Cyber-attack Detection based on three Models

## 1. Background information

Network attack detection is crucial in real life. The most obvious reason is security. Networks are constantly under threat from cyber-attacks that can lead to unauthorized access, data breaches, and various forms of cybercrime. So it is worthwhile to dive into cyber-attack detection methods. The cyber-attack detection problem is a typical classification problem, which can be effectively solved by machine learning techniques (Ahmad et al., 2022). Machine learning models are suitable for modelling complex relationships. Both traditional machine learning models and neural network models are used often in cyber-attack detection. Traditional attack detection methods attempt to model all kinds of attacks, which detect the attacks based on known knowledge. However, the relationships of network features are difficult to describe and represent. It is still a challenge to design a machine learning model that good at detecting cyber-attack.  
This report presents a comprehensive investigation into the application of three distinct machine learning models—Gradient Boosting Machines (GBM), Fully Connected Neural Networks (FCNN), and Long Short-Term Memory (LSTM) networks—in the detection of cyber-attacks. By exploring these models' capabilities and performance in classifying various types of network traffic, this research aims to shed light on the strengths and weaknesses of each approach. The ultimate goal is to evaluate their effectiveness in a real-world context, providing insights that can guide the development of more robust and efficient cyber-attack detection systems.

## 2. Data

a. Description

The CSV dataset comes from the website HackerRank. There are 39072 samples in train data and 4884 samples in validation data. The labels are the network traffic attack types of attack as the table below shows.

Tabel 2-1 Label and Descriptions

|  |  |
| --- | --- |
| Labels | Description |
| 0 | Normal traffic instance |
| 1 | Generic attack |
| 2 | Exploits attack |
| 3 | DoS attack |
| 4 | Fuzzers attack |
| 5 | Reconnaissance attack |
| 6 | Analysis attack |
| 7 | Backdoor attack |
| 8 | Shellcode attack |

b. Data preprocessing

The CSV dataset has 39 features, which are also the column names of the dataset. There is no need to extract features using other methods. However, features like 'src\_ip', 'dest\_ip', 'proto', 'state', and 'service' are not numeric, instead they are categorical. They need to be converted to numbers before putting in the models. I use frequency encoding to convert object features. Frequency encoding maintains the original number of features, preventing a blow-up in dimensionality, unlike one-hot encoding. It is a very simple and effective way to convert data.

There are also missing values in the dataset. I use the mean of each column value to impute the missing values. By implementing mean imputation, the distribution of the dataset remains. This action also retains the size of the dataset without dropping rows with missing values.

After processing the data, the task has become a multi-class classifier problem. Many machine learning models are good at classifying labels.

## 3. Build models

I choose the Gradient Boosting Machines (GBM) model, Fully connected neural network (FCNN) model and LSTM model to classify the type of network attack.

a. FCNN

I use Pytorch to implement the FCNN model. The FCNN model contains five fully connected layers. It maps 39 input features to 256 neurons, then sequentially reduces the dimension through layers to 128, 64, 32, and finally to 9 output classes. After each of the first four fully connected layers, there is a batch normalization layer, which helps to stabilize training and allows for higher learning rates. A dropout layer is defined with the dropout rate provided in the initialization. It is applied after the activation of each batch normalization layer to reduce overfitting. I choose Leaky ReLU as the activation function helps mitigate the issue of dead neurons. I also use hyperparameter tuning to find the best set of hyperparameters for the FCNN model.

b. GBM

For the GBM model, there are parameters like learning rate, number of estimators and maximum depth. I use grid search to find the best parameters and the scoring method is accuracy.

c. LSTM

I use Pytorch to build a four-layer LSTM model. Other hyperparameters such as learning rate, and batch size dropout rate are the same as the FCNN model.

## 4. Analyses results

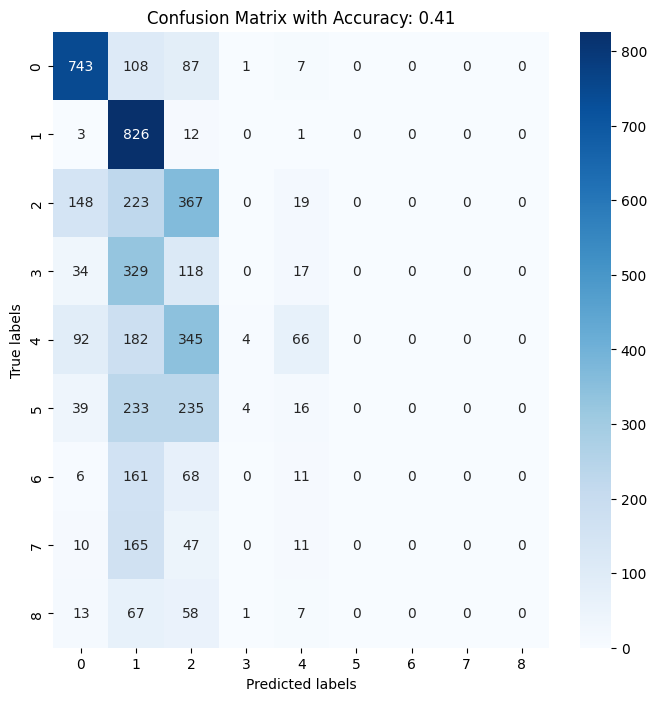
a. FCNN

Below is the confusion matrix of the FCNN model. The accuracy is 41%, which is modest. Considering the data has 9 labels, the random guess accuracy would be 11.1%. The FCNN model performs better than random guessing.

The model performs well in labels 0 and 1, which is also the majority class. From class 2, performance drops significantly, with a higher number of false positives and false negatives.

Classes 1 and 3 seem to dominate the predictions, suggesting that the model may be biased towards these classes. Although I use balanced class weight to offset the impact of imbalanced data in the training process, the result is still not good.

Several classes are commonly misclassified as classes 1 and 3. This could indicate that the features distinguishing these classes are not being effectively learned by the model, or that the classes are inherently difficult to differentiate based on the features provided.

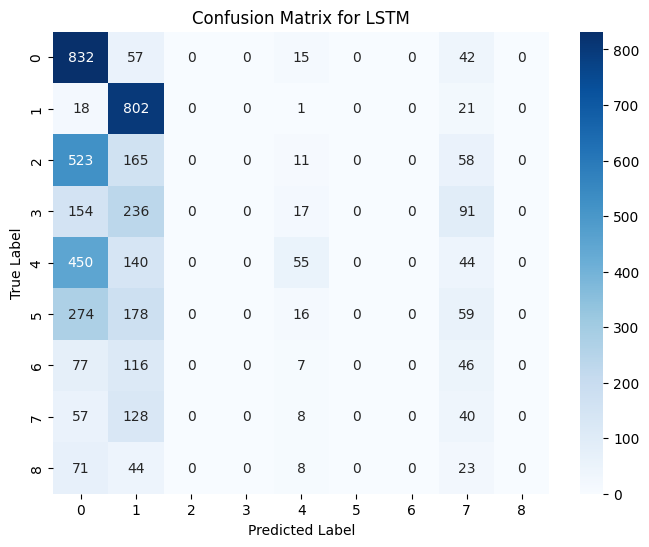


Picture 4-1 Confusion Matrix of FCNN

b. LSTM

Below is the confusion matrix of the LSTM model. The accuracy is 45.40%, which is lower than FCNN.

The model is performing well with classes 0 and 1, with high true positive counts. This suggests that the LSTM is effective at identifying these classes correctly. Class 2 has a high number of true positives, but there's a notable number of instances misclassified as class 1, indicating some confusion between these classes. For classes 3 to 8, the results are complicated. Class 3 has a relatively low true positive count compared to the number of false negatives, indicating a moderate performance. Class 4 has a similar issue with a true positive count of 450 but with a significant number of misclassifications as classes 1 and 3. Classes 5, 6, and 7 have more balanced misclassifications across several other classes. Class 8 has a low true positive rate and is often misclassified as Class 1. This suggests that the LSTM may be biased toward predicting class 1.



Picture 4-2 Confusion Matrix of LSTM

c. GRM

Below is the confusion matrix of the GBM model. The accuracy is 75.82%. It is higher than the accuracy of FCNN and LSTM, indicating the GBM model performed the best in this task.

For classes 0 and 1, the model is highly accurate for these classes with very few misclassifications. The model exhibits confusion between class 2 and classes 1 and 3. Class 4 shows a good true positive count, but there's notable confusion with class 3 and class 5. For classes 6 and 7, the model is confused with 5 and 8.

The diagonal of the matrix, which represents correct predictions, is dominant for most classes, which indicates that the model is generally performing well across the board.

日历

描述已自动生成

Picture 4-3 Confusion Matrix of GBM

## 5. Comparison

It is clear that all three models predict well on the majority classes. For the minority class, models are confused with features.

The FCNN model has moderate performance with notable misclassifications.

The LSTM model underperforms in this task. The dataset indeed does not have strong sequential patterns that the LSTM can leverage. This dataset has independent features.

In some network attack tasks that have time-series data, the LSTM can perform very well due to its internal gates regulating the flow of information (Yu et al., 2019). LSTMs are specifically designed to recognize patterns in sequential data, making them more effective for time-series analysis than models that treat all input data independently. In Fu and Yunshan’s research, when the architecture, which consists of data preprocessing, feature extraction, training, and attack detection, is tailored to process traffic data in networks, the LSTM model performs better than traditional machine learning models(Fu et al., 2018). By using LSTM in their approach, the researchers were able to design an end-to-end system that effectively classifies network traffic as normal or attack-related. The model was trained using the NSL-KDD dataset and reached a very high accuracy of 97.52%.

Although I have preprocessed the data and used balanced weight, there might be more nuanced relationships in the features that aren't being captured. Two neural network models are not overfitting considering the test loss is still higher than validation loss. This could involve interactions between features that a neural network might not be able to learn effectively. Also, even using tuning, the architecture might not be suitable. Maybe for this task, more layers are needed to capture the patterns. The choice of activation function may also not be suitable for this task. I choose Adam as the optimizer for FCNN and LSTM. Adam adjusts the learning rate for each parameter individually, which can lead to more efficient training and convergence. However, some studies have shown that models trained with Adam may not generalize as well as those trained with SGD and momentum, particularly on certain types of problems or datasets (Keskar & Socher, 2017). The paper also mentions that Adam often outperforms SGD in both training and generalization metrics in the initial training period but then performance plateaus. In contrast, SGD shows more consistent learning throughout the training process, which results in better generalization performance in the later stages. In my training process, it is true that the accuracy remains the same after a certain number of epochs. While Adam can accelerate the initial phase of training due to its adaptability and momentum-like properties, it might lead to solutions that don't generalize well because of the unique way it scales gradients for each parameter.

The GBM model shows significantly better performance, indicating a strong fit for the data characteristics. The balanced misclassifications and high true positives across most classes suggest that GBM effectively captures the data's underlying patterns. There are several reasons for its good performance. To begin with, with proper tuning of parameters like tree depth, learning rate, and the number of trees, GBMs can be less prone to overfitting (Ridgeway, 2007). The dataset in this task is tabular data with rows and columns, which GBM often excel in such structured data. The relationship between features can be highly non-linear but relatively straightforward to model with decision trees. What is more, unlike deep learning models, GBMs don't require explicit feature engineering to understand the interactions within the data. They are good at capturing complex interactions between features automatically (He et al., 2019). The training data have 39072 examples, which may not be enough for deep learning models to learn the patterns. In such cases, GBMs can outperform neural networks by making the most of the limited data.

## 6. Conclusion

The performance of each model can vary significantly depending on the nature of the dataset and the specific configuration of the model. In this task, while LSTMs and FCNNs have their advantages, especially in handling unstructured data and capturing complex, high-level abstractions, don’t perform very well. The imbalance between the distribution of different classes in the data and the choice of optimizer affect their ability to learn. GBMs perform better in this task as they are more effective for structured, tabular data where the relationships between features can be modelled through hierarchical decision rules.

Reference:

Ahmad, T., Truscan, D., Vain, J., & Porres, I. (2022). Early detection of network attacks using deep learning. *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 30–39.

Fu, Y., Lou, F., Meng, F., Tian, Z., Zhang, H., & Jiang, F. (2018). An Intelligent Network Attack Detection Method Based on RNN. *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 483–489. https://doi.org/10.1109/DSC.2018.00078

He, Z., Lin, D., Lau, T., & Wu, M. (2019). Gradient boosting machine: A survey. *arXiv Preprint arXiv:1908.06951*.

Keskar, N. S., & Socher, R. (2017). *Improving Generalization Performance by Switching from Adam to SGD* (arXiv:1712.07628). arXiv. http://arxiv.org/abs/1712.07628

Ridgeway, G. (2007). Generalized Boosted Models: A guide to the gbm package. *Update*, *1*(1), 2007.

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, *31*(7), 1235–1270.