Leonell Puso
ID# 302164083
CSC 180-01 Intelligent Systems (Fall 2023)

# PROJECT 2 REPORT

## (1) Problem Statement:

In this project, the objective is to develop a network intrusion detector, a predictive model capable of distinguishing between malicious connections (intrusions or attacks) and benign normal connections. The problem is formulated as a binary classification task, and we will evaluate different models' performance in terms of recall, precision, and F1-score for both attacks and normal connections. The following machine learning models will be employed:

- Logistic Regression (scikit-learn)
- Support Vector Machine (scikit-learn)
- Fully-Connected Neural Networks (TensorFlow)
- Convolutional Neural Networks (TensorFlow)

## (2) Methodology

**Data Collection:**

The UNSW-NB 15 dataset wil used, which reflects modern low-footprint attacks. The dataset comprises of nine types of attack categories, including Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. It includes a total of 49 features with a class label. Records are categorized as either 0 for normal connections or 1 for attack instances.

Note: Upon further inspection, the provided “cleaned” version by the professor are broken into training and testing set, which includes only 45 features.

**Data Cleaning:**

- Remove all the records with categorical values that only appear in training or test data
- Drop any rows with missing values
- Drop column that will not be used

**Feature Preparation:**

- Encode categorical features and normalize numeric features.

**Model Building:**

- Use training data to train your models and evaluate the model quality using test data
- Employ machine learning models using both scikit-learn and TensorFlow
- scikit-learn
    - Logistic Regression
    - Support Vector Machine
- TensorFlow
    - Fully-Connected Neural Networks
    - Convolutional Neural Networks

**Hyperparameter Tuning:**

- Find best number of neurons per input and hidden layers
- See how many hidden layers that will yield best result
- Apply activation functions relu, sigmoid, and tanh
- Use optimizers adam and sgd

**Training and Validation:**

- Use EarlyStopping and ModelCheckpoint to detect and prevent overfitting. Use this technique in five iterations but only keep the best model.

Evaluation and Analysis:

- Evaluate the model's performance by plotting the confusion matrix and ROC curve
- Compare the recall, precision and F1-score of the models for attacks and normal connections

## (3) Experimental Results and Analysis

For all FCNN model:

- 3 Hidden Layers
    - Neuron counts: first layer has 100, second layer has 50, third layer has 25

| **Fully-Connected Neural Networks** | | | |
|---|---|---|---|
| | relu | sigmoid | tanh |
| adam | Accuracy = 0.85<br>ROC = 0.84 | Accuracy = 0.84<br>ROC = 0.82 | Accuracy = 0.86<br>ROC = 0.85 |
| sgd | Accuracy = 0.84<br>ROC = 0.83 | Accuracy = 0.83<br>ROC = 0.82 | Accuracy = 0.84<br>ROC = 0.82 |

For all CNN models:

1. Convolutional Layer: 64 Neurons & Kernel Size: (1, 3)
2. MaxPooling Layer: Pool Size: (1, 2)
3. Flatten Layer
4. Fully Connected (Dense) Layer: 128 Neurons
5. Dropout Layer: Dropout Rate: 0.5
6. Fully Connected (Dense) Output Layer: 2 Neurons & Activation Function: Softmax

| **Convolutional Neural Network** | | | |
|---|---|---|---|
| | relu | sigmoid | tanh |
| adam | Accuracy = 0.85<br>ROC = 0.83 | Accuracy = 0.82<br>ROC = 0.80 | Accuracy = 0.87<br>ROC = 0.86 |
| sgd | Accuracy = 0.85<br>ROC = 0.84 | Accuracy = 0.82<br>ROC = 0.80 | Accuracy = 0.85<br>ROC = 0.84 |

## (4) Task Division and Project Reflection

Only one member.

**Project Reflection:**

This project was was deep dive into machine learning and neural network. For machine learning models, I learned more about logistic regression and support vector machines. In neural networks, Fully-Connected Neural Network and Convolutional Neural Network was worked on. Each of them had its unique way of tackling binary classification, and it was fascinating to see their strengths and weaknesses.

During the preprocessing I had a hard time figuring out how to make sure that both the training and testing dataset has matching categotical values. Using the function unique() helped a lot. Also, to make the data readable be the models, inputs for all the categorical features were encoded and the numeric features normalized.

Hyperparameter tuning was time consuming. Finding a sweet spot took some time. I experimented with different number of layers with different number of neurons. And I got to see how distinct set of activation and optimizer can make a difference in the models.

To avoid overfitting EarlyStopping and ModelCheckpoint were used. This made the models run for quite a long time. I had to lessen some of the complexity so that I am able to finish on time.

Finally, I got to see how well my models did. Metrics like recall, precision, F1-score, ROC curves, and confusion matrices to see how well the models did on predicting the outcome.

Looking back, this project was like a crash course in machine learning and neural network. I learned that getting your data right, picking the right models, fine-tuning them, and keeping them in check during training are the keys to success. It was an eye-opener, and I'm walking away with some serious skills.