Enhancing Intrusion Detection Systems through Advanced Machine Learning Techniques

Author: Rafael E. Gonzalez Cartagena
Advisor: Dr. Alfredo Cruz
Polytechnic University of Puerto Rico (M.S. Computer Science)

Abstract

The rise in cyberattacks during the Covid – 19 pandemic has intensified the need for intrusion detection systems. This study investigates three machine learning algorithms: Random Forest Classifier, Convolutional Neural Network – Focal Loss and Convolutional Neural Network – Cross Entropy Loss, under two project development lifecycles. Using the NSL – KDD Dataset, which contains network connection records classified into normal traffic or a multitude of different attacks, each of system is evaluated across multiple data handling staged, including scaling, balancing and feature selection. Results indicate that the Random Forest Classifier is best for binary classification, no matter the project development approach used (F1 Score: 99.92% in both cases); whereas Convolutional Neural Network – Cross Entropy Loss works best for multi – class classification when under the micro – detectors approach (F1 Score: 99.59%). As such, the findings of this research would offer practical guidance for designing machine learning systems for intrusion detection systems.

Introduction

The rapid shift to remote work and online digital services triggered by the Covid – 19 pandemic has been accompanied by a noticeable rise in cyberattacks, placing a renovated interest on Intrusion Detection Systems (IDS). Traditional IDS technologies, commonly based on signature – based detection, anomaly – based detection and stateful protocol analysis, struggle to keep up with increasingly sophisticated attacks and evasion techniques, some of which leverage on Artificial Intelligence (AI) themselves. At the same time, however, AI, specifically, Machine Learning (ML), offers a promising way to enhance IDSs performances by identifying and learning patterns of both normal and anomalous network activity directly from data inputs.

Since the start of the pandemic, organizations around the world have experienced roughly a 50% increase in weekly cyberattack incidents [1], with ransomware being the most common threat and averaging $2.75 million in damages per event as of 2025 [2]. Despite this trend, most research comprising of both ML and IDSs has focused specifically on deep learning models trained under the traditional, monolithic approach, where a single ML system is built to recognize every class in a target label. As such, little attention has been given to alternative project development lifecycles, such as the micro – detectors approach, in which a multitude of model instances are trained and evaluated per target class to potentially improve such a model's overall adaptability and performance [3].

This study addresses those gaps by comparing three ML systems, these being: Random Forest Classifier, Convolutional Neural Network (CNN) – Focal Loss and CNN – Cross Entropy Loss, under both the monolithic and micro – detectors approaches, using the NSL – KDD Dataset. Specifically, this work seeks to answer the following questions:

1. What constitutes artificial intelligence, machine learning, and intrusion detection systems?
2. What types of data – centric attributes do the NSL – KDD Dataset offer for IDS development?
3. What are the key architectural and methodological distinctions between the monolithic and micro – detector approaches for ML design?
4. Do ML models developed under the micro – detector approach demonstrate superior performance compared to those which use the monolithic alternative?
5. Based on the findings of prior research, is it correct to assume that deep learning algorithms, like convolutional neural networks, will always outperform traditional methods?

Methodology

This study employs the NSL – KDD Dataset for all experiments, with a designed data handling pipeline, two contrasting project development lifecycles, i.e.: monolithic and micro – detectors; three machine learning algorithms, and a combination of performance evaluation metrics.

The NSL – KDD Dataset is an improved successor to the original KDD CUP 1999 Dataset, containing 148,517 entries and 43 columns, where 29 and 14 are numeric and categorical, respectively. Its target column is called ATTACK_LABEL, and comprises 40 distinct computer network attack classes, like Neptune, ipsweep, satan, smurf, nmap and mailbomb. The dataset also has a new column not found in its predecessor, namely: DIFFICULTY_CLASS, in which each record classification difficulty is rated on a 1 – 21 scale. Notwithstanding, in the case of the used training set, 390 duplicated records, a 0.30% of the set itself, were identified. Table 1 shows the names and descriptions of a subset of the columns found in the NSL – KDD Dataset [4].

Table 1: NSL – KDD Columns Descriptions

Column Name	Description
DURATION	Length, in seconds, of the network connection.
PROTOCOL_TYPE	Type of network protocol that was used.
SERVICE	Type of network service that was used.
FLAG	Status flag, that is that was used during the network connection. In other words, it signals how the TCP handshake will proceed.
SRC_BYTES	Number of data bytes from source to destination.
DST_BYTES	Number of data bytes from destination to source.
LAND	Indicator of whether the source IP address and source port are equal to the destination IP address and destination port, respectively. A *1* indicates that such values are the same; otherwise, a *0* is used.
WRONG_FRAGMENT	Number of out of order fragments throughout the network connection.
URGENT	Number of urgent packets sent.
HOT	Number of administrative indicators in a given packet payload. Examples of indicators include file access, file creation and file deletion commands.
NUM_FAILED_LOGINS	Number of failed login attempts.
NUM_COMPROMISED	Number of security incidents.
ATTACK_LABEL	Label indicating connection type. *Normal* indicates that the connection is secure, while anything else represents the opposite, an *Attack Category*.

Additionally, Figure 1 shows the correlation matrix of the dataset, where it can be seen that WRONG_FRAGMENT, LAND and LOGGED_IN are the three features that correlated the most with ATTACK_LABEL.

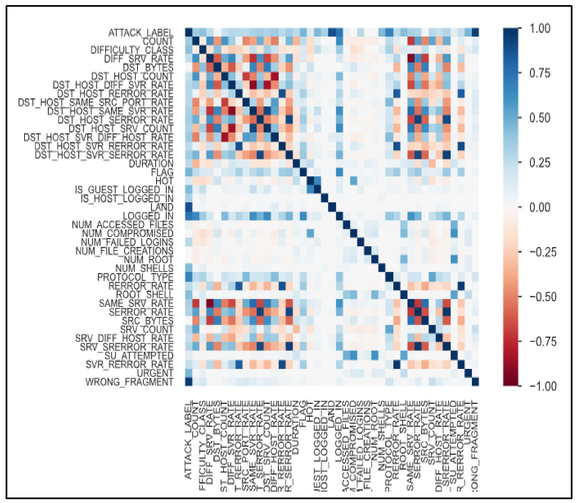

Figure 1: NSL – KDD Correlation Matrix

The data handling pipeline of this study works as follows: (1) Basic Preprocessing, where NSL – KDD CSV files are loaded into Pandas DataFrames for training and testing, categorical columns are identified and converted into one – hot encoded versions using Scikit – Learns OneHotEconder, and binary, binary one – hot encoded, and five – class integer and one – hot encoded (Normal: 0, DoS: 1, U2R: 2, R2L: 3, Probe: 4) of the ATTACK_LABEL column are created; (2) Feature Scaling, which is comprised of the application of Scikit – Learn's StandardScaler, to all features in both the training and testing sets; (3) Balancing, where Synthetic Minority Over – Sampling Technique (SMOTE) is executed on the training set only, generating a multitude of synthetic samples of the minority classes until all classes are uniform to one – another; (4) Feature Selection, in which the top 85% of features are selected for each model via the usage of a Mutual Information Classifier, L1 – Regularized Linear Regression (LASSO), and Random Forest Classifier. The mathematical equations for Scikit – Learn's StandardScaler function, as well as those of the Mutual Information Classifier and LASSO are shown respectively below:

$$\sigma = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N}} \qquad I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

$$F_{objective} = (B_0 + B_1x_1 + B_2x_2 + \ldots + B_nx_n + \varepsilon) + \lambda \times (|B_1| + |B_2| + \ldots + |B_n|)$$

Notwithstanding, regarding the project development lifecycles, all models were trained and tested with:

- Monolithic Approach: A single model is trained to classify all target labels found in the dataset.
- Micro – Detectors Approach: Each class found in the target label is treated as a separate detector, or a model instance. In this case, for all tested ML systems, their performance metrics are the average of the performances of all their instances.

Likewise, with regards to the ML systems themselves, for the Random Forest Classifier, Scikit – Learn's RandomForestClassfier class was used. In the case of the CNNs, PyTorch was used instead, with the implementation or understanding of the Loss Functions Cross Entropy Loss and Focal Loss. Cross Entropy, defined below, is a CNN criterion designed to evaluate the difference between the predicted probability distributions and the true, label distributions. Focal Loss is an alternative to Cross Entropy Loss; however, it is useful for class imbalance.

$$CE(y,\hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y_i}) \qquad FL(p_t) = -a_t(1-p_t)^{\gamma}\log(p_t)$$

And, finally, the model evaluation metrics that were used throughout this research are:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

There are also the Receiver Operating Characteristic Area Under the Curve (ROC AUC) that is used to break ties in the F1 Scores of models, and execution time.

Results and Discussion

The performances of all three models are evaluated under both the monolithic and micro – detectors approaches with 2 and 5 target label classes. Considering those that performed the best under the F1 Score metric, random forest classifiers are best suited for binary classification tasks regardless of the methodology used (F1: 99.92% / 99.92%), whereas CNN – Cross Entropy is best for multiclass classification tasks whenever micro – detectors is used (F1: 99.59%). Figure 2 shows the best performances of these models.

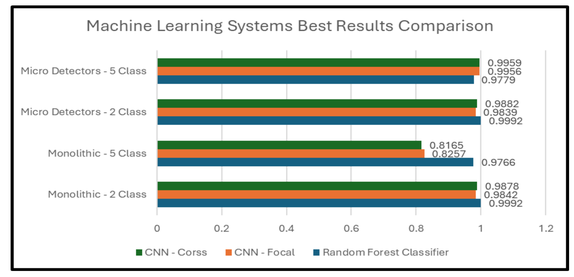

Figure 2: Best Results Comparison

Moreover, with regards to execution time, the Random Forest Classifier constantly outperformed the CNNs with time – ranges of 139.63 – 1,762.30 seconds, shown in Figure 3.

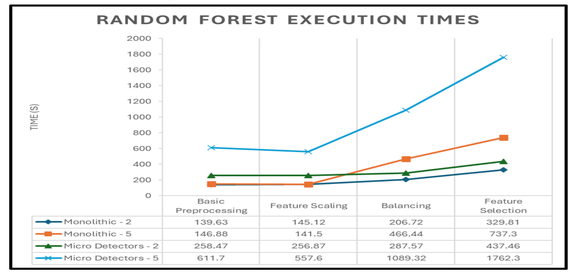

Figure 3: Random Forest Classifier Execution Times

Conclusions and Future Work

This study shows that Random Forest Classifier excels in binary tasks, while CNNs perform well for multiclass when proper preprocessing is applied. With feature scaling, balancing, and selection, the monolithic five-class Random Forest Classifier achieved 99.86% accuracy, surpassing prior work [5]. Contrary to Zhao et al., CNNs did not consistently outperform RFC [6]. Future work includes extending the codebase with new techniques and evaluating additional machine learning algorithms.

Acknowledgements

This material is based upon work supported by Dr. Alfredo Cruz and Professor Edna Chaar.

References

[1] BuisnessWire, "Cyber Threats Have Increased 81% Since Global Pandemic," 9 November 2021. [Online]. Available: https://www.businesswire.com/news/home/20211108005775/en/Cyber-Threats-Have-Increased-81-Since-Global-Pandemic.

[2] R. Sobers, "Ransomware Statistics, Data, Trends, and Facts [updated 2024]," 13 November 2024. [Online]. Available: https://www.varonis.com/blog/ransomware-statistics.

[3] S. Saad, W. Briguglio and H. Elmiligi, "The Curious Case of Machine Learning In Malware Detection," 2019. [Online]. Available: https://arxiv.org/abs/1905.07573v1.

[4] Y. SAHLI, "A comparison of the NSL-KDD dataset and its predecessor the," International Journal of Scientific Research and Management (IJSRM), 2022.

[5] Revathi and Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning," 12 Decmber 2013. [Online]. Available: https://www.ijert.org/research/a-detailed-analysis-on-nsl-kdd-dataset-using-various-machine-learning-techniques-for-intrusion-detection-IJERTV2IS120804.pdf.

[6] F. Zhao, H. Li, K. Niu, J. Shi and R. Song, "Application of deep learning-based Intrusion Detection System (IDS) in network anomaly traffic detection," 2024. [Online]. Available: https://www.ewadirect.com/proceedings/ace/article/view/15060.