Network Intrusion Detection System Using Anomaly Detection Techniques

David Oroian
Communications Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Aalto University
Espoo, Finland
david.oroian@aalto.fi

Roland Bolboaca
Department of Engineering and Engineering Technology
"George Emil Palade" University of Medicine,
Pharmacy, Science and Technology of Targu Mures
Targu Mures, Romania
roland.bolboaca@umfst.ro

Adrian-Silviu Roman
Department of Engineering and Engineering Technology
"George Emil Palade" University of Medicine,
Pharmacy, Science and Technology of Targu Mures
Targu Mures, Romania
adrian.roman@umfst.ro

Virgil Dobrota
Communications Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
virgil.dobrota@com.utcluj.ro

***Abstract*—In the current digital landscape, protecting networks against malicious activities is a critical challenge. Network Intrusion Detection Systems (NIDS) are vital as the first line of defense, continuously monitoring network traffic to detect and prevent potential attacks in real-time. As cyber-attacks get more complex and more similar to normal traffic, robust NIDS solutions have become more crucial than ever. This paper proposes an architecture for an anomaly-based NIDS that has a Multi-Layer Perceptron (MLP) as a binary classifier. The system employs the Cisco TRex generator to simulate network traffic, capturing and analyzing the data using tcpdump and Zeek, and lastly preprocessing it for the MLP using Python scripts. Three algorithms were evaluated for the classification task: Isolation Forest (IF), MLP, and Autoencoder, all implemented with TensorFlow and Keras. The models were trained and tested on two widely recognized datasets for anomaly detection, KDDCUP99 and UNSW-NB15. The experimental results show the superiority of the UNSW-NB15 dataset compared to the KDDCUP99 one in terms of complexity and its likeness to real-world traffic. Moreover, the results also prove that a simple Deep Learning (DL) algorithm such as the MLP can serve as an effective first-line defense against cyber threats. This study contributes to the ongoing development of more effective NIDS by exploring the application of machine learning techniques in anomaly detection, offering the potential for enhancing network security.**

***Keywords*—Anomaly Detection, Autoencoder, Cyber Security, Isolation Forest, KDDCUP99, MLP, NIDS, UNSW-NB15**

I. Introduction

Intrusion Detection Systems (IDS) are categorized into three main types: Signature-based (SIDS), Anomaly-based (AIDS), and hybrid systems [1]. SIDS rely on known attack patterns but struggle with zero-day attacks, while AIDS can detect new threats by identifying deviations from normal behavior. However without careful algorithm selection, proper model optimizations, and an adequate thresholding methodology, they often suffer from high false positive rates. Hybrid systems combine SIDS and AIDS to enhance accuracy and recall [2, 3], though they may still face challenges with false positives.

Signature-based, often referred to as Knowledge-based or Misuse Detection [2], imply that there is a database of known patterns of attacks to which the system matches its given inputs. This method has the disadvantage of not being able to identify zero-day attacks, and it requires significant effort to update the database with new attack patterns as they occur.

Anomaly-based systems have the advantage of identifying zero-day attacks, as they learn the normal pattern of data and, by analyzing the deviation of new data from the normal pattern, decide whether or not an anomaly was encountered. Their disadvantage is the high rate of false positives. Such a system was developed in [2] using an AdaBoost classifier ensemble made up of a Decision Tree (DT), Naive Bayes (NB), and Artificial Neural Network (ANN). The results reported an increased performance in terms of accuracy and recall. The downside of this method is the result of significant overhead [3].

The hybrid systems combine both SIDS and AIDS architectures in order to improve the precision and recall of the model. Such a technique was proposed in [3], using a C5 DT classifier for the SIDS and a One-Class SVM with a Radial Basis Function (RBF) kernel for the AIDS. The SIDS checks to see if it can match the input traffic to a known attack signature from a database, if it cannot, the input goes further to the AIDS. The latter is trained with samples representing normal traffic and decides on how to classify the traffic based on a chosen threshold. If it gets classified as malicious, its signature is put into the database so it can be detected by the SIDS from now on. The combination of the two stages resulted in an overall accuracy of 94%, the false positive rate being the biggest downside to the model.

This study proposes an effective NIDS by investigating the application of machine learning techniques in anomaly detection, offering the potential for enhancing network security. The approach was tested and evaluated on three Machine Learning (ML) models for anomaly detection: IF, MLP and Autoencoder, on the KDDCUP99 and UNSW-NB15 datasets. For the features of the datasets, only a subset was used from each one, both taken from literature. We also created a testbed for real-time traffic classification for which the MLP model chosen and the features used in the KDDCUP99 dataset were extracted based on the comparison conducted.

The experimental results show the superiority of the UNSW-NB15 dataset compared to the KDDCUP99 one in terms of complexity and its likeness to real-world traffic. Moreover, the results also prove that a simple Deep Learning (DL) algorithm such as the MLP can serve as an effective first-line defense against cyber threats. This study contributes to the ongoing development of more effective NIDS by exploring the application of machine learning techniques in anomaly detection, offering potential pathways for enhancing network security.

The rest of this paper is organized as follows: Section II offers an overview of the current state-of-the-art in NIDS. Section III then details the reasoning of our chosen solution. Section IV provides a comprehensive description of the proposed approach, followed by the presentation of experimental results in Section V. Finally, the paper concludes in Section VI.

II. Related Work

A variant of an Anomaly Intrusion Detection System (AIDS) using an AdaBoost classifier ensemble was proposed in [2]. They have generated and evaluated the TCP/IP model, focusing on Message Queuing Telemetry Transport (MQTT), Domain Name System (DNS), and Hypertext Transfer Protocol (HTTP) protocols. In order to capture the traffic from their test-bed they have used tcpdump for capturing traffic in pcap files, the Bro-IDS tool in order to generate connection-based features from the generated pcap files, and at last using a custom extractor module to create the remaining features for the input of the model out of the features generated in the logs by the Bro-IDS tool. The connection-based features, which represent network flows, present an aggregated view of the network. It is more advantageous to analyze flows compared to any other datasets as the time spent analyzing them is considerably reduced [4]. In [2] the correntropy measure [5] was used to estimate the similarities between normal and abnormal features, showing a clear distinction between the two classes [2]. The results reported an increased performance in terms of detection rate and recall. The downside of this method is the result of significant overhead [3].

In [3], Khraisat et al. proposed a hybrid NIDS made out of an AIDS and a Signature-based Intrusion Detection System (SIDS), using a C5 DT classifier for the SIDS and a One-Class SVM with a Radial Basis Function (RBF) kernel for the AIDS. The two classifiers were combined using the boosting method, which also resulted in decreasing the bias of the system. The Hybrid Intrusion Detection System (HIDS) has the aim of detecting both patterns for known attacks and novel ones while at the same time reducing the false positive rates and increasing the detection rate. The dataset used for training and validating the model is the Bot-IoT dataset, which was created by Koroniotis et al. to simulate real Internet of Things (IoT) network traffic as well as possible attack types [6]. Feature selection, in this approach, was tackled using Information Gain (IG), because of its advantageous execution time which is crucial when it comes to IoT devices. The combination of the two stages resulted in an overall accuracy of 94%.

Another approach to AIDS was developed in [4] using a stacking approach. This method was first introduced as a concept in [7] and was later shown to produce more reliable results by the publication of the Super Learner article [8]. In [4] three classifiers were used as base models - K-nearest Neighbour (KNN), logistic regression (LR), and random forest (RF) - from which the metaclassifier, a Support Vector Machine (SVM), learns to predict the samples. The article underlines the fact that something hard to improve is the detection rate for attacks such as analysis, Denial Of Service (DOS), worms, and backdoors. The main advantage of this method is the improvement in the accuracy of the prediction when given as input unbalanced datasets [9].

III. Proposed Methodology

In this paper, which is based on our preliminary work from [10], we created a solution for detecting cyber threats, both known and novel ones, while packaging everything in a software which is easy to distribute on as many devices as possible. The first target was achieved by using an anomaly detection technique and the latter by containerization.

The proposed NIDS has the purpose of using an MLP classifier trained on available datasets for labeling generated traffic. It is composed of two main components, the TRex and Device Under Test (DUT) Docker containers. The TRex container is the one generating the network traffic and the DUT is capturing the traffic using tcpdump and extracting the needed flow-based features from it using the Zeek tool. These extracted features are further processed using a python script in order to generate the entire subset of features needed for the input of the MLP. In the end, the features are fed to the MLP trained on the UNSW-NB15 dataset and it classifies the generated data points as either normal or anomalies. The flow and architecture of the proposed method can be seen in Fig. 1. As it can be seen in the previously mentioned figure, two networks were used in order to send normal traffic on one of them and abnormal on the other. The traffic was generated using scripts provided by the TRex tool, namely stl/imix.py for the normal traffic and stl/syn_attack.py for the anomalies.

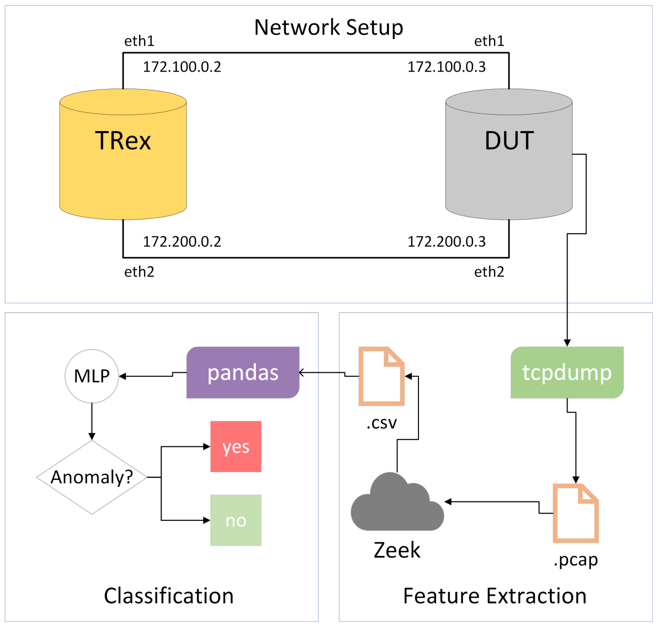


Fig. 1: Proposed testbed for NIDS.

A. Models

1) Isolation Forest: An unsupervised learning algorithm designed to identify outliers by uncovering the dataset's underlying structure. It consists of multiple tree structures that recursively divide the dataset based on its features. In each iteration, a feature is randomly selected, and the data is split using a randomly chosen value within the range of that feature.

This process continues until the entire dataset is divided, creating a single tree within the forest. Anomalous data points typically have much shorter paths from the root to a leaf compared to normal data points, as they are more easily isolated. All in all, the algorithm is very good at handling high-dimensional data, as it does not need more data points alongside the addition of features. It also is very efficient, therefore it scales very well to high data sizes. There are however some drawbacks, the model requires more splits when data points are very close to each other and it also has to be told what proportion of anomalies to expect. Also, even though the model gives an anomaly score, it is hard to tell how it got to it because of the large number of trees in the forest. The IF algorithm is a very good anomaly detector as long as the parameters are tuned right.

2) Multi-layer Perceptron: An MLP is the simplest form of a neural network. It represents a sequence of layers, each of them composed out of a collection of neurons computing an output (see 1) using the given input. The output of one layer is fed as the input of the next layer, with the first layer being the data we want to classify or make predictions out of and the last layer being the output of the network for the desired task. All the layers between the input and output ones are called *hidden layers*.

$$y = f(\sum_{i=1}^{n} x_i w_i + b) \tag{1}$$

The forward pass is the process during which data passes from the input layer to the output layer and the result is computed for the desired data. The backward pass refers to the process of moving through the network in the other way, such that during the process, the gradients are computed through backpropagation or *backward propagation of errors*. After computing the gradients, the learnable parameters are optimized. The gradients are a higher-extension of the derivative and the way they help at updating the parameters starts from the reasoning that going in the direction opposite to the gradient, we should arrive at the minimum of the *loss* function, the equation for the process is described in (2). The learning rate is the step size with which we update the parameter x in order to get to the minimum loss.

$$x = x - \text{learning rate} * \text{gradient} \tag{2}$$

There are other more complex optimizers than the simple gradient descent such as RMSProp and Adadelta, which dynamically adapt the learning rate per parameter and follow the typology of the loss curve better. They are optimal to use for high-dimensional problems or for situations in which the data is sparse. The most popular optimizer is the Adam optimizer, which, alongside RMSProp and Adadelta implements accumulated squared gradients and momentum, but in addition to the latter ones it also facilitates more stable learning through the incorporation of bias correction. That is why the Adam optimizer is usually the go-to optimizer across a wide range of problems. This being said, there is no perfect optimizer and for each problem it has to be checked which of the available optimizers perform the best, each of them having their own pros and cons.

3) Autoencoder: Autoencoders are neural networks designed to discover low-dimensional representations of high-dimensional data and to reconstruct the input out of this low-dimensional representation. They are made out of two pieces: an encoder and a decoder. The encoder has the task of reducing the dimensions of the data given as an input and the decoder must reconstruct the initial data as best as it can starting from the encoded set. This principle stands at the foundation of building an anomaly detection module. The general architecture of an autoencoder can be seen in Fig. 2.

The compression and decompression functions implemented by neural networks are lossy and, for the most part, unsupervised. The entire network is typically trained as a single unit. The loss function is generally either the mean-squared error or cross-entropy between the output and the input, referred to as the reconstruction loss, which penalizes the network for producing outputs that differ from the initial data. Since the encoding (the output of the central hidden layer) is significantly smaller in dimension than the input, the encoder must decide what information to discard. The encoder learns to retain as much relevant information as possible within the

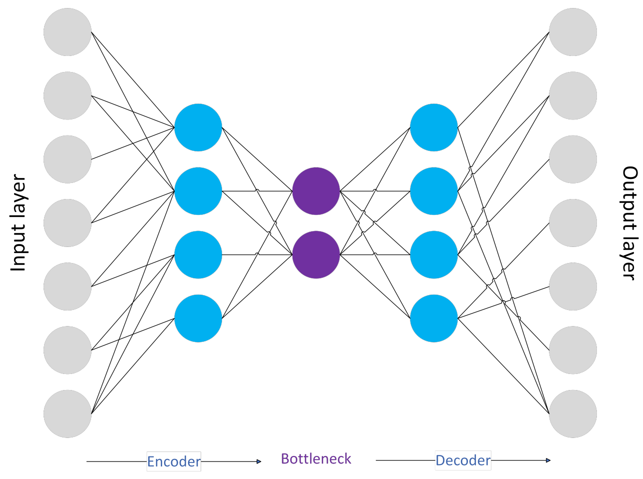


Fig. 2: Autoencoder architecture.

limited encoding, discarding the irrelevant parts intelligently. The decoder, in turn, learns to reconstruct the input accurately from the encoding. Another way of constraining the representation besides the size of the hidden layers is the addition of a sparsity constraint on them, such that in the end, fewer units are active at any given moment. The model which results from this procedure is called a sparse autoencoder.

IV. Experimental Assessment

A. Datasets

An important part of developing a NIDS is training the models on relevant datasets. One of the most used datasets in literature, the KDDCUP99 dataset has been developed by Stolfo et al.[11] using the captures made during the DARPA'98 IDS program. The dataset was comprised of nine weeks in which raw tcpdump files were generated. The set consists of 41 features of 4,900,000 connection vectors, labeled as an attack or as normal traffic. Each attack record is labeled as one of four attack types: Denial of Service (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L), and Probing attack [12].

A more robust dataset, which better simulates real traffic and cyber attacks, is the UNSW-NB15 dataset created by Moustafa and Slay[13]. This dataset presents a comprehensive blend of realistic network activities and synthetic attack behaviors. Generated using IXIA PerfectStorm and captured with tcpdump, it comprises 100 GB of network traffic data. The traffic was generated from three virtual servers, two of them generating normal traffic and the third one generating abnormal traffic which was taken according to the CVE site to produce a real modern thread environment. Feature extraction, facilitated by Argus and Bro-IDS tools, yielded 49 features across 2,540,044 records in four CSV files. The dataset categorizes attack types into nine groups, including Fuzzers, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, and Worm. These classifications offer insights into various network threats.

B. Data analysis and preprocessing

We started by analyzing the data points by looking at the distribution of the labels in the datasets. The task we tried to solve was one of binary classification, that being determining whether a point is either 0, if the data point is normal, or 1, in the case it is anomalous. We converted all of the attack labels to the values of 1 and the normal traffic to 0 and the result for KDDCUP99 can be seen in Fig. 3. Here we can see that the number of anomalies surpasses the normal data points four times.

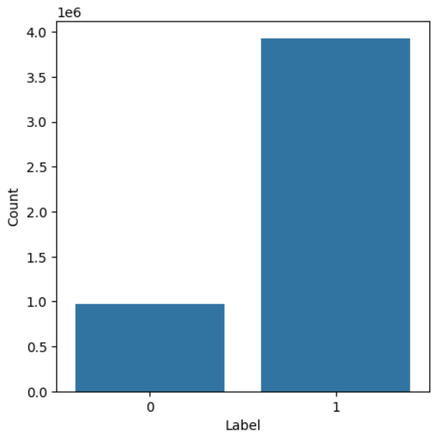


Fig. 3: Distribution of normal(0) and anomalous(1) data for KDDCUP99

For the UNSW-NB15 dataset, which already has the test and train sets split, the results can be seen in Fig. 4 and Fig. 5. This set is almost balanced while the train one has a distribution of roughly 2:1 for anomalies with respect to normal points.

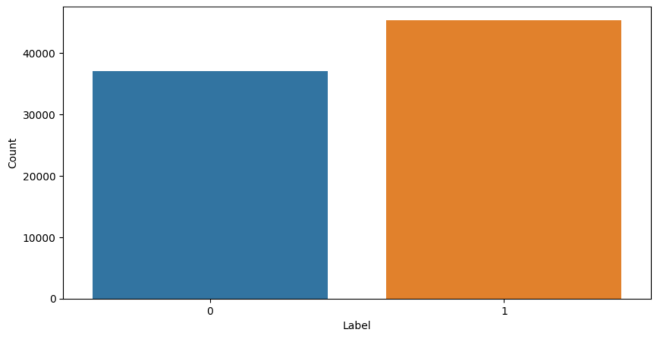


Fig. 4: Distribution of normal(0) and anomalous(1) data for UNSW-NB15 test set.

We tested both under and oversampling techniques to balance the datasets. We concluded that for the tuned models, whether the sets were balanced or not, made no difference in terms of performance.

For both the datasets and for all of the algorithms used, We had to encode the categorical variables into numerical values and this was done using the *LabelEncoder* from the scikit-learn library.

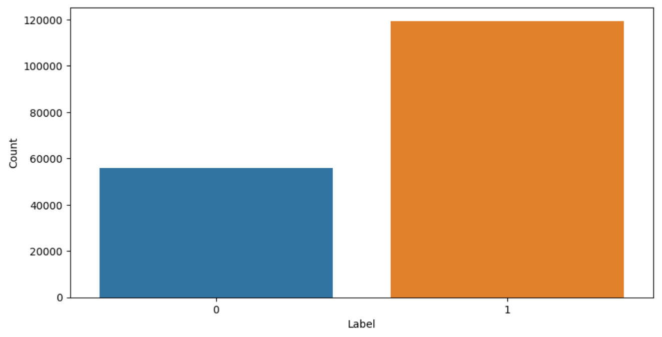


Fig. 5: Distribution of normal(0) and anomalous(1) data for UNSW-NB15 train set.

C. Train, test, validation splitting

As the UNSW-NB15 dataset already has pre-split test and train sets, We only split the test set into test and validation ones, with a 70/30 ratio. The test data was split on both datasets only for the MLP and Autoencoder models as the IF one does not use the validation split. For the KDDCUP99 dataset, the train-test split ratio was 80/20 with the test set being further split as previously explained.

D. Tuning

For hyperparameter tuning, we have chosen the *ParameterGrid* provided by scikit-learn for iterating through different combinations of parameters and the *cross_val_score* also provided by scikit-learn for conducting the validation of the parameters [14]. The validation procedure is a k-fold cross-validation one, which is a thorough and robust strategy as it partitions the dataset into k bins (determined by the *cv* parameter) iterating through them with each one being the testing set while the rest are the training one.

E. Isolation Forest

For the IF model, the parameters we achieved the best results with for the KDDCUP99 dataset were:

- number of estimators = 100(the number of base estimators/trees in the forest)
- contamination = 2e-2(the proportion of anomalous data in the trained set)
- Max features = 0.8(% of features to draw from the dataset to train each base estimator)

For automatically determining the threshold above which points are considered anomalies, we trained a Logistic Regression (LR) model on the computed average path lengths (anomaly scores) and the actual outputs. The LR has the same equation as the sigmoid function (3).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

As we will later see, the IF algorithm was not able to successfully classify the UNSW-NB15, no matter how we tuned it, because it was not able to separate normal and anomalous points in terms of average path lengths.

F. Autoencoder

For the KDDCUP99 the optimal parameters for the autoencoder are:

- batch size = 64
- epochs = varied due to EarlyStopping
- intermediate dimensions = 8, 4
- latent dimension (bottleneck) = 2
- hidden layers activation function = ReLU
- output layer activation function = sigmoid
- hidden layers kernel initializer = he_uniform
- kernel regularizer = L2
- optimizer = Adam with MAE loss function

The autoencoder was used in order to compute the error between the reconstructed inputs and the actual inputs and determine which points are anomalies or not based on this error [15]. For computing the error, the Mean Squared Error (MSE) (4) was computed between the inputs and reconstructions after which we computed the Median Absolute Deviation (MAD) score of the errors described in [16] as (5) where $\hat{Y}$ is the median of the data. This results in a more optimal way of computing the errors [16].

Similarly, as for the IF model, the LR model was used to automatically determine the threshold separating anomalies and normal points. And even with LR added, as in the case of the IF, the Autoencoder was not able to separate the points as normal or anomalous for the UNSW-NB15 dataset.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y_i})^2 \tag{4}$$

$$\text{MAD} = median(|Y_i - \hat{Y}|) \tag{5}$$

G. MLP

For the MLP, the optimal model had the following parameters for the KDDCUP99 dataset:

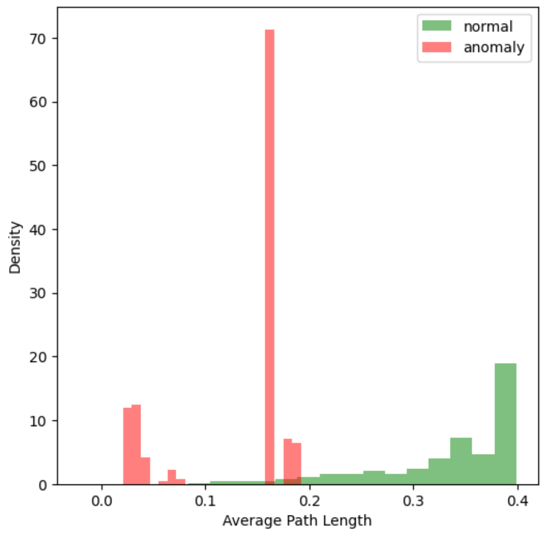
- batch size = 64
- epochs = varied due to EarlyStopping
- number of hidden layers = 2
- number of neurons per hidden layer = 32
- hidden layers activation function = ReLU
- output layer activation function = softmax
- hidden layers kernel initializer = he_uniform
- kernel regularizer = L2
- optimizer = Adam with 1e-3 learning rate and categorical cross-entropy loss function

The only difference between the model for the KDD set and the UNSW-NB15 one being the kernel initializer for the hidden layers which in this case was chosen to be the glorot_uniform one.

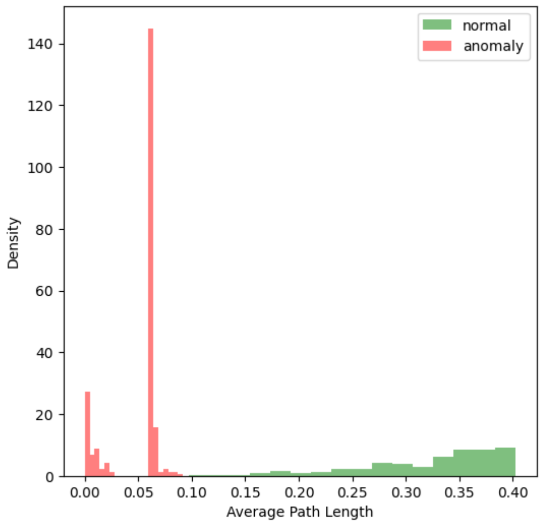
H. Performance metrics

For talking about performance metrics we have to define the terms forming a confusion matrix:

- True Positive (TP): Predicted output and real one are positive



(a) No feature selection



(b) Feature selection

Fig. 6: Comparison of normalized distribution of average path lengths for Isolation Forest on the KDDCUP99 set.

- True negative (TN): Predicted output and real one are negative
- False negative (FN): Predicted output is negative and real one is positive
- False positive (FP): Predicted output is positive and real one is negative

$$\text{F1-measure} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6}$$

Using these terms we can define the following metrics used to evaluate the models in this paper: Accuracy, which is defined by (7), precision defined by (8), and recall as (9). The latter is also known as True Positive Rate (TPR).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{9}$$

It is important to note that accuracy as a standalone is not the best descriptor of a model. A better choice is the F1 measure as in order to have a good F1 score, the model needs to have a good precision as well as a good recall score. The formula for the F1-measure can be seen in (6).

V. Results and Discussion

The results of the methods proposed in this paper can be seen in comparison to the literature in Table I.

A. Isolation Forest

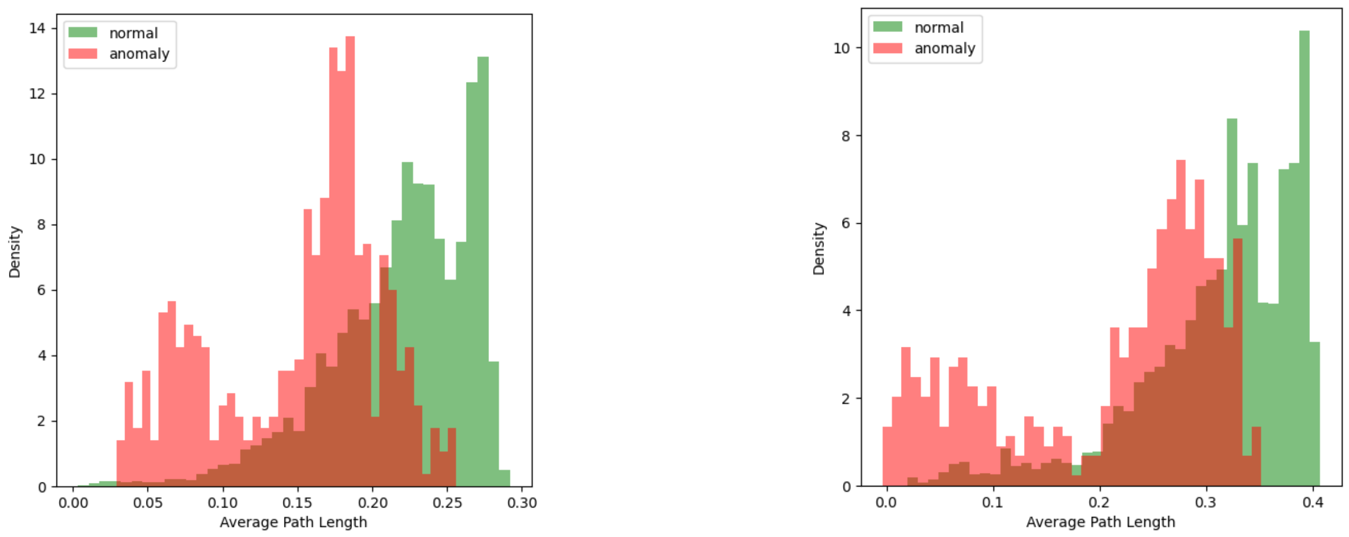
1) KDDCUP99: For this dataset, on the IF forest algorithm, it can be seen how important feature selection can be in the context of the performance of a model. Its impact on the performance of this specific model is highlighted in the distribution of the average path lengths for anomalies and normal points of the model trained on feature-selected data next to the model trained on the whole dataset. The comparison is shown in Fig. 6. It can be clearly seen from the figure that, first of all, the anomalies have a smaller average path length than normal points which is as they should. Secondly, when no feature selection is applied the clean and anomaly points overlap each other around the value 0.18 in terms of average path length determined by the IF model. On the other hand, when feature selection was applied, the model was better at separating the normal from the anomalous points.

The results of this model classifying the test set are $F1 = 95.06\%$, $accuracy = 99.79\%$, $precision = 91.2\%$, $recall = 98.97\%$. This is a very good result, especially looking at the recall and accuracy of the model which are very high as they are not affected by the FP (the accuracy is but very slightly due to the big imbalance in the normal points and anomalies). And even though there are some FP results, the model can be considered as very satisfactory in terms of performance.

2) UNSW-NB15: If the Isolation Forest algorithm alongside the LR could well classify the data points for the KDDCUP99 dataset after it has been tuned and feature selection was done, it is not the case for the algorithm classifying the UNSW-NB15 data. Looking at the distribution of the path lengths for the model trained with and without feature selection (Fig. 7) this time we can note that, firstly, for the case with no feature selection the normal and anomalous data points are indistinguishable from one another and this does not improve

TABLE I: Literature NIDS results compared to the proposed approach.

Author	Model	Dataset	no. of features	Feature selection method	Performance (accuracy, precision, recall)
Pu et al. [17]	ACO-SVM	KDDCUP99	Not mentioned	Not applied	**-, 99.2%, -**
Ma et al. [18]	BPSO-SVM	KDDCUP99	variable size	BPSO	**99.44%, - , -**
Rajagopal et al. [4]	Stacked Classifier	UNSW-NB15	11	IG and hashing	**94%, 96%, 93%**
Lin et al. [19]	CANN	KDDCUP99	6	Not applied	**99.76%, -, -**
			19		**99.46%, -, -**
Hassan et al. [20]	LSTM	UNSW-NB15	Not Mentioned	CNN	**97.17%, -, -**
Lv et al. [21]	ELM+DE+Gravitational Search	UNSW-NB15	Not mentioned	PCA	**89.01%, - , -**
		KDDCUP99			**96.59%, -, -**
Khammassi and Krichen [22]	DT	UNSW-NB15	variable size	GA-LR	**92.7%, - , -**
		KDDCUP99			**99.4%, -, -**
Waheed Ali H. M. et al. [23]	HADMLP	KDDCUP99	variable size	MOABC	**97.2%, - , -**
Yin et al. [24]	MLP	UNSW-NB15	23	IGRF-RFE	**84.24%, - , -**
Ghanem and Jantan [25]	EBAT-MLP	KDDCUP99		Not applied	**91.1%, - , -**
		UNSW-NB15			**96.86%, - , -**
This paper	MLP	UNSW-NB15	11 [4]	Not applied	**94.68%, 84.12%, 97.21%**
	MLP	KDDCUP99	15 [26]		**99.85%, 99.96%, 99.76%**
	IF	KDDCUP99			**99.79%, 91.2%, 98.97%**
	Autoencoder	KDDCUP99			**99.84%, 94.3%, 92.8%**



(a) No feature selection

(b) Feature selection

Fig. 7: Comparison of normalized distribution of average path lengths for Isolation Forest on the UNSW-NB15 set.

much when conducting the feature selection on the dataset.

B. MLP

1) KDDCUP99: The results for this model are $F1 = 99.86\%$, $accuracy = 99.85\%$, $precision = 99.96\%$, $recall = 99.76\%$. These numbers are from a 10-run average of the results. The model was trained also on the whole dataset besides the balanced one which is present in the confusion matrix, but there were no notable differences. The performances for this model are almost perfect, which had to be double-checked, but there is no leaky information in the process. It is to be noted that many literature results surpass 99% accuracy.

2) UNSW-NB15: For this the metrics are $F1 = 90.19\%$, $accuracy = 94.68\%$, $precision = 84.12\%$, $recall = 97.21\%$. The MLP was able to classify the data from the UNSW-NB15 dataset in a satisfactory manner, the only downside of the model being the somewhat high FP rate, which we were not able to compensate through any of the tuning that we have done to the model. This result underlines the improved ability of Neural Network models to find connections between data points that cannot be found using traditional Machine Learning models.

C. Autoencoder

The results for the autoencoder for the KDDCUP99 set can be seen in Table I. This is once again a good result compared with the literature. As for the UNSW-NB15 set, the model was once again, as in the case of the IF, not able to coherently detect the anomalies.

VI. Conclusion

In this paper, we proposed an architecture for an anomaly-based NIDS and validated the model's performance using well-known datasets. The proposed models demonstrated the ability to classify network traffic into anomalies and normal points. The results were consistent with other models in literature.

However, of the three models proposed, only one produced coherent results for the UNSW-NB15 dataset, highlighting the dataset's more complex structure and its closer simulation of real-world scenarios. In future work, we aim to integrate a SIDS into the architecture to enhance traffic classification efficiency. While we utilized the TRex traffic generator in these experiments, it was less versatile than anticipated. Consequently, exploring other solutions like IXIA PerfectStorm, known for generating close to real traffic and more robust attack scenarios, would be a valuable objective. Moreover, we plan to compare our IDS with an existing and publicly available software on the market.

Acknowledgment

An initial expanded version was presented by David Oroian as B.Sc. in Telecommunications thesis at Technical University of Cluj-Napoca in July 2024.

References

[1] A. Pinto, L.-C. Herrera, Y. Donoso, and J. A. Gutierrez, "Survey on Intrusion Detection Systems Based on Machine Learning Techniques for the Protection of Critical Infrastructure," *Sensors*, vol. 23, no. 5, p. 2415, Feb. 2023.

[2] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.

[3] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks," *Electronics*, vol. 8, no. 11, p. 1210, Oct. 2019.

[4] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets," *Security and Communication Networks*, vol. 2020, pp. 1–9, Jan. 2020.

[5] W. Ma, H. Qu, and J. Zhao, "Estimator with forgetting factor of correntropy and recursive algorithm for traffic network prediction," in *2013 25th Chinese Control and Decision Conference (CCDC)*. Guiyang, China: IEEE, May 2013, pp. 490–494.

[6] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, Nov. 2019.

[7] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[8] M. J. Van Der Laan, E. C. Polley, and A. E. Hubbard, "Super Learner," *Statistical Applications in Genetics and Molecular Biology*, vol. 6, no. 1, Jan. 2007.

[9] K. Kerwin and N. D. Bastian, "Stacked Generalizations in Imbalanced Fraud Data Sets using Resampling Methods," Apr. 2020.

[10] D. Oroian, "Anomaly-Based Network Intrusion Detection System Using Machine Learning Techniques," BSc, Technical University of Cluj-Napoca, Jul. 2024.

[11] S. Stolfo, Wei Fan, Wenke Lee, A. Prodromidis, and P. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2. Hilton Head, SC, USA: IEEE Comput. Soc, 1999, pp. 130–144.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. Ottawa, ON, Canada: IEEE, Jul. 2009, pp. 1–6.

[13] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*. Canberra, Australia: IEEE, Nov. 2015, pp. 1–6.

[14] M. A. H. Zamrai, K. M. Yusof, and M. A. Azizan, "Random Forest Stratified K-Fold Cross Validation on SYN DoS Attack SD-IoV," in *2024 7th International Conference on Communication Engineering and Technology (ICCET)*. Tokyo, Japan: IEEE, Feb. 2024, pp. 7–12.

[15] A. Sridhar and K. A. Suman, *Beginning Anomaly Detection Using Python-Based Deep Learning*. APress, 2024.

[16] "Detection of Outliers," National Institute of Standards and Technology, 2024, https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h.htm.

[17] J. Pu, L. Xiao, Y. Li, and X. Dong, "A Detection Method of Network Intrusion Based on SVM and Ant Colony Algorithm," in *Proceedings of 2012 National Conference on Information Technology and Computer Science*. China: Atlantis Press, 2012.

[18] J. Ma, X. Liu, and S. Liu, "A New Intrusion Detection Method Based on BPSO-SVM," in *2008 International Symposium on Computational Intelligence and Design*. Wuhan, China: IEEE, Oct. 2008, pp. 473–477.

[19] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, Apr. 2015.

[20] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Information Sciences*, vol. 513, pp. 386–396, Mar. 2020.

[21] L. Lv, W. Wang, Z. Zhang, and X. Liu, "A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine," *Knowledge-Based Systems*, vol. 195, p. 105648, May 2020.

[22] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Computers & Security*, vol. 70, pp. 255–277, Sep. 2017.

[23] G. Waheed Ali H. M., E.-E. Yousef A. Baker, A. Mohamed, T. Mohammad, A. Nayef A. M., N. Abdullah B., A. Nibras, and A.-w. Ola A., *Metaheuristic Based IDS Using Multi-objective Wrapper Feature Selection and Neural Network Classification*. Singapore: Springer Singapore, 2021, pp. 384–401.

[24] Y. Yin, J. Jang-Jaccard, W. Xu, A. Singh, J. Zhu, F. Sabrina, and J. Kwak, "IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset," *Journal of Big Data*, vol. 10, no. 1, Feb. 2023.

[25] W. A. H. M. Ghanem and A. Jantan, "A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm," *Neural Computing and Applications*, vol. 32, no. 15, pp. 11 665–11 698, Aug. 2020.

[26] L. Li, H. Zhang, H. Peng, and Y. Yang, "Nearest neighbors based density peaks approach to intrusion detection," *Chaos, Solitons & Fractals*, vol. 110, pp. 33–40, May 2018.