ALFA BK UNIVERSITY
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
FACULTY O F COMPUTER SCIENCE
ALFATECH
Proceedings

NEURAL NETWORK-POWERED INTRUSION DETECTION SYSTEM

*Đorđe Karišić[1]; Milan Čabarkapa[2]**

Abstract

This paper provides a comprehensive exploration of an approach addressing the dynamic challenges posed by modern cyber threats through the integration of neural networks into intrusion detection systems. The given approach emphasizes the necessity for adaptive defences in the face of ever-evolving threats, which camouflage as normal network traffic. The methodology, meticulously detailed in the paper, defines the importance of constructing such a system that adeptly identifies underlying patterns in the network data.

Key words: *Neural Networks, Intrusion Detection Systems, Cybersecurity, Machine Learning, Adaptive Defence Mechanisms*

[1] Đorđe Karišić, University of Kragujevac, Faculty of Engineering, Sestre Janjić 6, 34000 Kragujevac, Serbia, djordje00karisic@gmail.com

[2] Milan Čabarkapa, University of Kragujevac, Faculty of Engineering, Sestre Janjić 6, 34000 Kragujevac, Serbia, mcabarkapa@kg.ac.rs *

Introduction

Traditional “Intrusion Detection Systems” often struggle to keep pace with the dynamic nature of cyber threats and modern malicious software, which aims to disrupt selected networks, resulting in critical errors or stolen information.

Malicious actors strategically disguise their activities within legitimate data transmissions. Oftentimes, such activities occur posing as normal network traffic, making their detection challenging for the traditional “Intrusion Detection Systems”. The camouflage within the ordinary network flow allows these threats to go unnoticed, due to their similarities with the normal network data, resulting in the risk of critical errors and potential exposure of sensitive information.

Traditional “Intrusion Detection Systems” rely on predefined set of rules to convey and conduct a series of operations in order to detect and prevent such malicious activities exposing the network’s information and flow of data. This approach may struggle to differentiate between benign and malicious activities, when the latter adopts the disguise of former. Dynamic nature of cyber threats demands an adaptive and proactive, defensive solution.

This paper explores the pressing need for a more sophisticated approach to network intrusion detection, recommending the integration of neural networks with “Intrusion Detection Systems”, aiming to enhance their capabilities to recognize potential anomalies which expose the presence of malicious activities. By integrating neural networks, which are inspired by human brain’s ability to learn and to adapt [1], “Intrusion Detection Systems” gain an ability to learn from and adapt to the given network data, ensuring detection of deviations from the norm and system’s evolution. Evolution of the system’s understanding of the processed data, enables system’s continuous progression of understanding what constitutes normal network behaviour and data flow.

Augmentation of the anomaly detection capabilities of detection systems, enables this approach to seek to uncover and neutralize potential risks that lurk behind the cover of normal network traffic, ensuring a robust, defensive mechanism in real-time against the ever-evolving adversary.

This paper navigates through the steps of exploring and developing such approach, discussing the architecture, neural network methodologies, and integration of real-time threat intelligence. Through a comprehensive exploration of the system’s capabilities and real-world performance metrics, the provided methodology aims to showcase how this innovative approach keeps pace with the dynamic nature of cyber threats, especially those that manifest as seemingly normal network traffic. Neural networks [2] represent a paradigm shift in the field of cybersecurity, as they offer a unique and adaptive solution to the intricate challenges posed by modern cyber threats. This integrated approach marks a significant step towards fortification of digital defences against the ever-evolving cyber threats.

Materials and Methods

This section explores the steps involved in dataset exploration and preparation, feature engineering and definition of the recommended approach for intrusion detection.

Dataset

The dataset which was employed in this study is *NF-ToN-IoT* dataset [3]. This dataset consists of raw network traffic data, derived from diverse networked environments, specifically curated for the purpose of advancing research in intrusion detection and network security. *NF-ToN-IoT* contains *1,379,274* data flows, of which *1,108,995* (80.4%) are malign and *270,279* (19.6%) are benign.
Each instance of network traffic data listed in this dataset consists of features concisely described in Table *1*: Features of each dataset instance.

Table 1: Features of each dataset instance

Feature	Description
IPv4 Source Address	Source IP address from which the network traffic originates.
Source Port Number	Source port number associated with the network traffic.
IPv4 Destination Address	Destination IP address to which network traffic is directed.
Destination Port Number	Destination port number associated with the network traffic.
Protocol	Network protocol used for communication.
Layer 7 Protocol	Application layer protocol.
Number of incoming bytes	Defines the volume of data received.
Number of outgoing bytes	Defines the volume of data transmitted.
Number of incoming packets	Measures the count of received packets.
Number of outgoing packets	Measures the count of transmitted packets.
TCP flags	Flags associated with TCP communication.
Duration of network flow	Duration of network flow in milliseconds.
Label	Binary label indicating the nature of the network traffic (0 – benign, 1 – malign)
Attack	Categorical label specifying the type of attack (e.g., DDoS)

Each and every feature, represented as a column in the dataset, exhibits complete data integrity [3], meaning that there are no missing values describing the feature of an instance. This ensures reliability and dataset completeness, resulting in seamless further analysis.
The set of features, as given in Table *1*, represented via columns, provides system with the essential characteristics of each instance of data traffic, enabling it to evolve by finding correlation between values of different features within the given set, therefore learning the importance of each feature and how to interpret their values intelligently.

Further dataset discussion includes statistical analysis of class distribution and visual comparison. Each categorical value, such as type of the attack and label are used to provide and process the distribution of instances. Visual comparison is given by histograms represented by Figure *1*: Distribution of instances - benign and malign and Figure *2*: Distribution of instances - types of network attack.

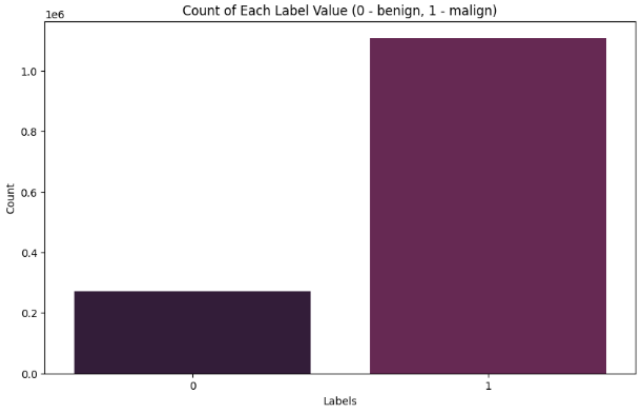

Figure 1: Distribution of instances - benign and malign

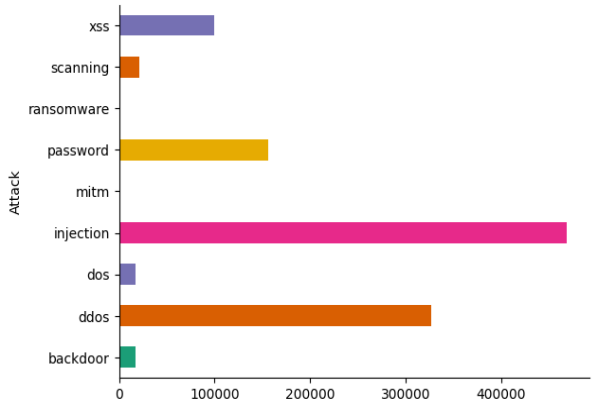

Figure 2: Distribution of instances - types of network attack

Examination of the histogram provided by Figure *1*: Distribution of instances - benign and malign results in further insight into the distribution of the instances, in this case, distribution of malign and benign data flows. This histogram suggests that there lies a potential challenge posed by the class imbalance, which may impact the neural network model's ability to accurately identify and classify minority instances (benign data flows).
Histogram represented by Figure *2*: Distribution of instances - types of network attack also suggests that class imbalance exists in the distribution of the types of network attacks.
The context of this paper is exclusively focused on the binary classification of data flows as either benign or malign.
The solution to the class imbalance problem, as suggested by this paper, is to dynamically adjust the learning process through the incorporation of class weights during model training. By assigning appropriate weights to each class, the neural network becomes adept at discerning subtle pattern within both benign and malign instances. This adjustment ensures that the model

allocates more significance to the underrepresented class (benign) without compromising its ability to correctly classify instances from the majority class (malign).
As mentioned beforehand, correlation between each of the features proves to yield critical information about the state of the dataset and its instances; therefore, must be analysed comprehensively.
Figure 3: Features - Correlation Matrix is a visual representation of the correlation matrix made from the features in the *NF-ToN-IoT* dataset. This correlation matrix represents the relationships between the features, providing an overview of the interdependencies within the dataset.

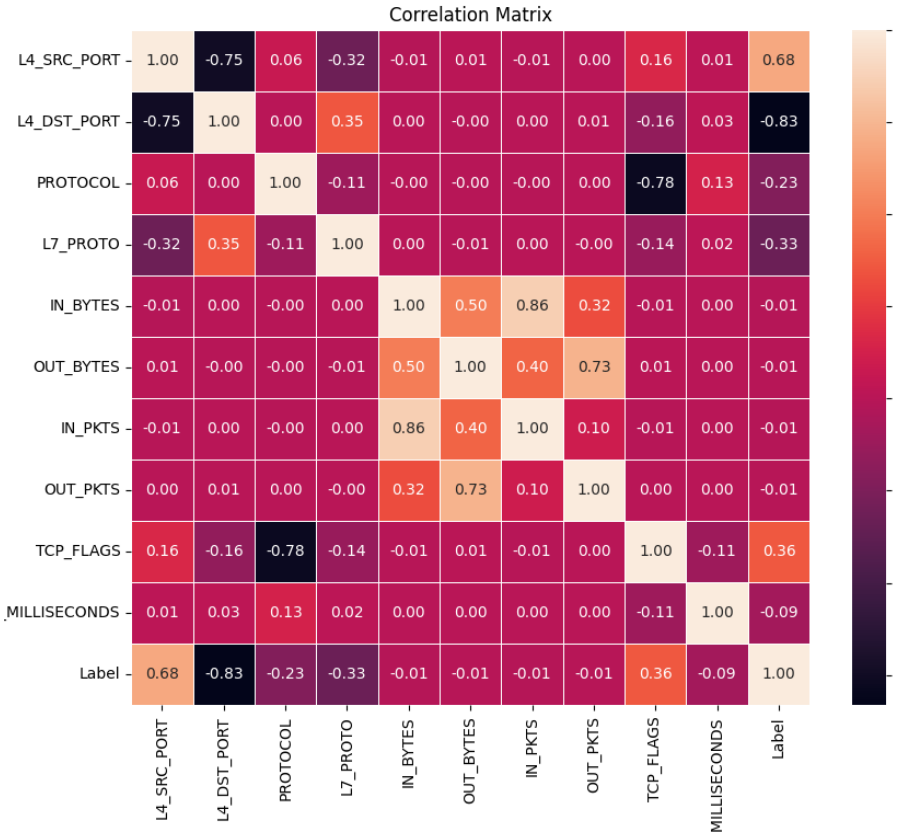

Figure 3: Features - Correlation Matrix

Correlation matrix utilizes a colour gradient, ranging from shades of black to beige, to indicate the strength of correlation and its coefficient. Shades of red represent that there's no correlation between features. Ideally, these values should be close to zero, signifying a lack of correlation. Several conclusions could be derived from this correlation matrix:

- Correlation between *Source port* and *Destination port* numbers features suggests a somewhat strong negative correlation.
- Correlation between *Label* and *Destination port* number, too, suggests somewhat strong negative correlation.
- Correlation coefficient of *-0.78* between *Protocol* and *TCP Flags* features indicates that when the former's value increases, latter's value tends to decrease, resulting in somewhat strong negative correlation.

Absence of strong correlations among all other features enforces the significance of each feature's unique contribution. Retaining the minority of somewhat strongly correlated features enforces model to discern subtle differences between instances, ensuring it can adapt to the dynamic nature of the cyber threats.

Methodology

Methodology, as defined in this paper, outlines the synthesis of a *Neural Network* for an "Intrusion Detection System", preceded by a sequential chain of operations, which contributes to data preprocessing and preparation.
The goal is to develop an adaptive, robust system with a neural network at its core, capable of recognizing and reacting to subtle patterns found within the dataset – contributing to potential enhancement of any network's security against cyber threats.
As mentioned previously, complete system architecture consists of multiple elements and operations, from which three main elements are singled out:

- **Input** – set of instances described by their features, classified as either benign or malicious data flow.
- **System** – set of internal procedures, variables, methods and processes tasked with preparing, managing and processing given input data to ensure seamless, efficient and precise training, validation and performance evaluation of the internal neural network.
- **Output** – tuple of elements containing information regarding performance evaluation of the training phase (*validation accuracy*) and testing phase (*classification report*).

Methodology of this paper suggests a specific pipeline of operations to be constructed prior to feeding the neural network data from the input. Suggested pipeline consists of an ordered list of operations and procedures which all connect given input to the neural network, conducting required preprocessing. All operations within this pipeline are sequentially connected, meaning that one operation's output is another operation's input, in a given order.

1. **Data Cleaning and Selection Operation** – Ensures that input data is represented in via specifically selected features, and that none of them have rows with missing values, or contain unsupported values.
2. **Preprocessing IP Addresses –** Naturally, IP addresses are represented as a set of four numbers, each in range from 0 to 255. As such, they cannot be processed correctly and meaningfully by the neural network in the same format. One way of preprocessing them is to convert them to strip the dots and add leading zeros to ensure each octet contains exactly three digits (e.g. an IP address represented in standard format as 192.168.10.1. could be represented as a 64-bit signed integer 192,168,010,001). This way ensures that the relationships between IP addresses are maintained, meaning that neural network model can potentially recognize adjacent nodes, devices and networks.
3. **Data Scaling** – Previous item in this list suggested the approach of conversion of IP addresses so they can be represented as 64-bit signed integers. As the dataset defined in this paper contains a large number of instances, of which each contains two IP addresses (*Destination and Source*), it is very important to consider the computational demands of the neural network. One way of reducing the computational demands is to apply a scaling technique. One such technique is called **min-max scaling** [4] (*min-max normalization*) and is defined by equation
4.

5. $X_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$ (1).

$$X_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Application of min-max scaling on each of the numerical features, ensures that the computational demands of the neural network get optimized, as data values lie within constant, smaller ranges [4].

6. **Dataset Split** – To ensure proper training, validation and testing phases, dataset needs to be split, effectively and intelligently, into three different subsets, for each phase. Dataset is split using 70:15:15 ratio, which allocates 70% of data to the training set and 15% to both validation and test sets. Stratification of the data ensures that each subset maintains the same distribution of class labels, preserving the balance between benign and malign instances.
7. **Calculation of Class Weights** – Class weights are computed to assign different weights to each class, ensuring that the neural network appropriately adapts to instances from the minority class. Calculated class weights are forwarded to the neural network, to assist in calculation of the loss function [5].

The neural network architecture deployed with the task of classifying *NF-ToN-IoT* dataset instances is carefully designed, as to capture the most intricate of details. The architecture consists of a few key components, each contributing to the model's abilities and adaptability.

1. **Input Layer** – Used to predefine and essentially prepare the network to work with specific shapes of input data.
2. **Processing Layers** – Multiple dense layers, followed by batch normalization [6] and dropout layers are incorporated in the architecture, to enable and improve model's ability to learn and capture patterns, and reduce overfitting by introducing regularization techniques.
3. **Activation Functions** – Rectified Linear Unit (*ReLU*) activation functions [7] are included after each dense layer, increasing the complexity and adding non-linearity within the neural network. This enables neural network model to learn more complex representations of the data.
4. **Output Layer** – Utilizes sigmoid activation function to produce a value in the range 0 to 1 representing binary classification. This value indicates the likelihood of an instance being benign or malign data network flow.
5. **Model Compilation** – Model is compiled using the Adam optimizer [8] and binary cross-entropy loss function.

Visual representation of the given model's architecture is presented by *Figure 4*: Neural Network Architecture *and Figure 5*: Neural Network Architecture ex. 2.

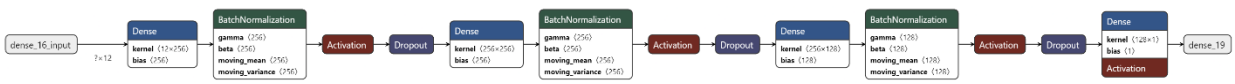

Figure 4: Neural Network Architecture ex. 1

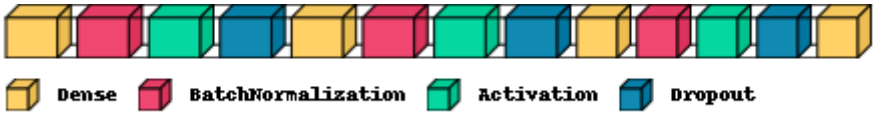

Figure 5: Neural Network Architecture ex. 2

Neural network's hidden layers architecture incorporates three sequentially connected blocks which follow the same block construction rules – "*Dense*" layer followed by "*Batch Normalization*", "*Activation*" and "*Dropout*" layers.
Output layer is presented with a single "*Dense*" layer containing one neuron with sigmoid activation function.

The neural network training phase incorporates several internal processes and elements, which ensure proper handling and parametrization of the training phase.

1. **Number of epochs** – Each epoch represents passing the entire dataset forward and backward through the neural network once. Number of epochs is set to **100** for the purposes of the training outlines by this approach.
2. **Batch Size** – Number of instances processed together before the update of neural network's parameters [6]. Batch size is set to **256**, due to the large number of network flow instances.
3. **Callback for Learning Rate Reduction** – Learning rate influences the size of adjustments made to the model weights based on the calculated gradients. This callback ensures dynamic reduction of learning rate, enhancing and speeding up model's convergence.
4. **Callback for Early Stopping** – Callback which prevents unnecessary computations and therefore overfitting. After **7** unsuccessful epochs, training phase is terminated and weights from the model's best performing epoch are selected and restored as they encapsulate model's optimal state achieved in the training phase, for further model's performance evaluation.

Complete system recommended and projected by the approach to the methodology given by this paper is presented by *Figure 6*: Complete methodology, encapsulating the entire approach for effective neural network training.

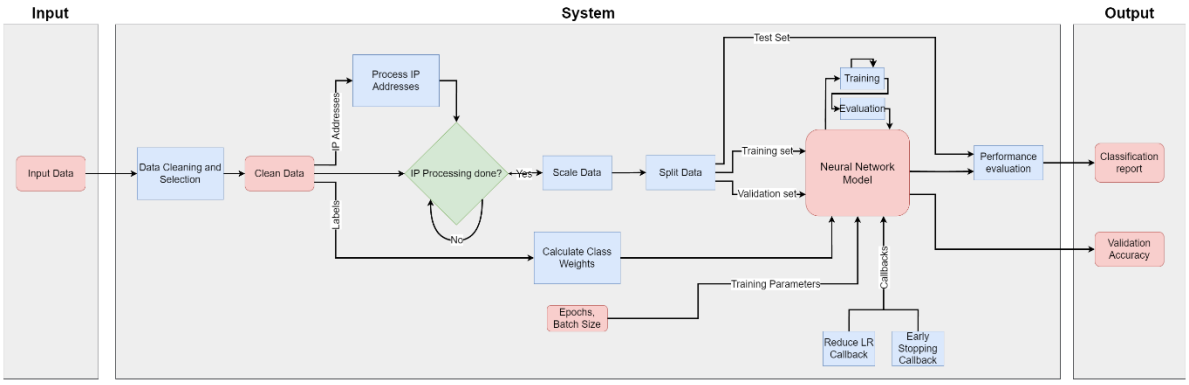

Figure 6: Complete methodology

The carefully designed methodology integrates multiple components to ensure correct and effective neural network's training and evaluation in case of network intrusion detection. The strategic combination of parameters such as the number of epochs, batch size and callbacks ensure both effective and performance-oriented training phase. Defined approach, outlined by *Figure 6*: Complete methodology, presents the complete system's framework and pipelines ensuring a robust and adaptive neural network capable of addressing the given cyber threats.
After establishing the proposed framework for effective neural network training, it is advised to shed some light on the achieved results and neural network performance evaluation.

Results Discussion

Proper evaluation of network's performance requires performing an analysis that covers various aspects of its predictive capabilities, and generalization to new, unseen data. Such predictive capabilities can be put to test by utilizing the test set. Feeding the deployed model with test set data results in generation of predicted class labels, which will be subjected to further analysis by comparing them with the original class labels using different evaluation metrics.
Figure 7: Confusion Matrix - Test Setshows a confusion matrix [9], representing *True Negatives*, *False Negatives*, *True Positives* and *False Positives*, in a matrix-like structure, mapping each of them in a clockwise manner.

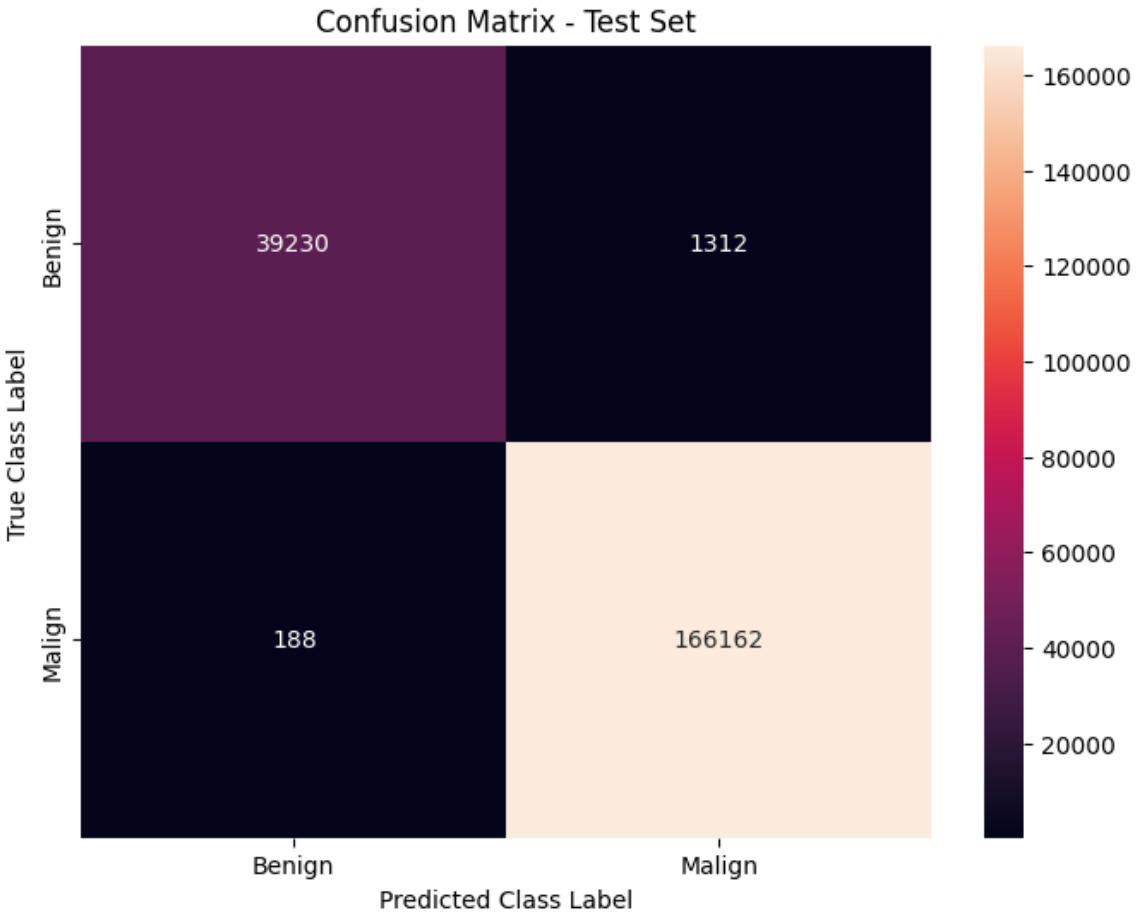

Figure 7: Confusion Matrix - Test Set

From the confusion matrix and its elements, given in *Figure 7*: Confusion Matrix - Test Setmultiple different evaluation metrics arise, *Precision*, *Recall*, *F1-Score* and *Accuracy* [9]. Each of them is utilized to describe model's performance at predicting specific class label utilizing a different approach. Numerically represented evaluation results are given for each of class labels or combined as average and weighted average results of each applied evaluation metric.

Table 2: Evaluation metrics and values presents a set of evaluation factors, calculated from confusion matrix's elements.

Table 2: Evaluation metrics and values

Evaluation metric [9]	**Benign ("0")**	**Malign ("1")**	**Average**	**Weighted Average**
Precision	99.52%	99.22%	99.37%	99.28%
Recall	96.76%	99.89%	98.33%	98.33%
F1-Score	98.12%	99.55%	98.84%	99.27%
Number of samples used	40542	166349	206891	206891
Accuracy (Overall)	99.27%			

Results suggest almost perfect model performance. Further analysis requires delving deeper into the model's performance, analysing its classification threshold. Normally, classification threshold is set to 0.5, but to conduct a more thorough analysis of the model's behaviour, it is essential to explore its performance across a range of different thresholds. This analysis provides insight into the trade-off between precision and recall, and stability and confidence of the model. **Threshold Sensitivity Analysis** of the model's performance on **malign class labels** will yield enough information to make an informed conclusion, provided by *Figure 8:* Threshold Sensitivity Analysis of malign class labels

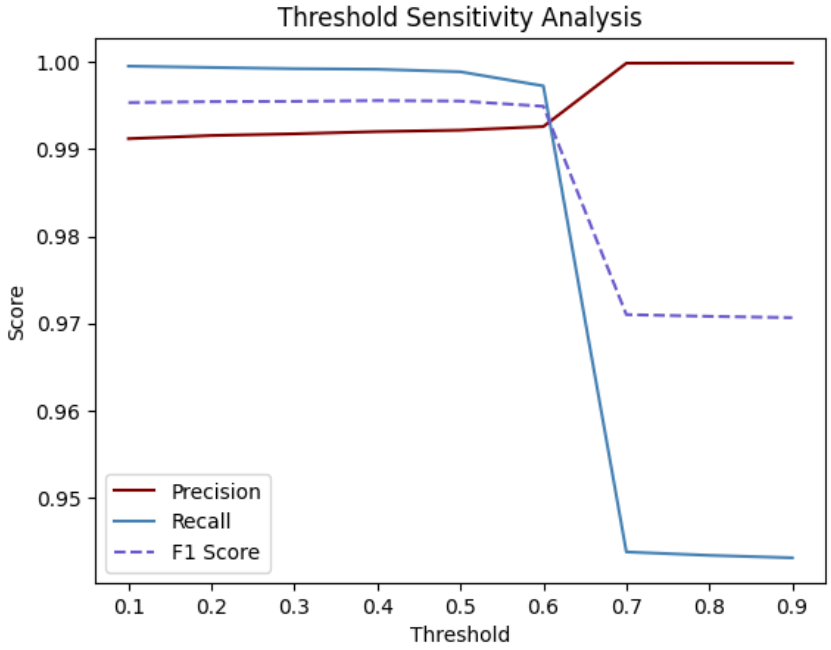

Figure 8: Threshold Sensitivity Analysis of malign class labels

Several occurrences could be observed from the *Figure 6.*

- **Recall Decrease** – As threshold increases, model becomes less sensitive in capturing all instances. It is making more false negative predictions.
- **Precision Increase** – As threshold rises, model becomes more selective in labelling instances as malign. It is making fewer false positive predictions.
- **Conclusion** – Higher threshold increases precision at the expense of decreasing recall.

Conclusion

This paper presented a comprehensive exploration of a neural network-powered approach to solving the problem of network intrusion detection. The dynamic nature of cyber threats indicates a need for an adaptive approach to their elimination, thus the integration of neural network offers a promising solution.

The examination of the given, developed approach, as listed in methodology section, defines the importance of careful construction and synthesis of system that can precisely and effectively address the underlying patterns that occur in day-to-day network traffic and threats that might appear. The recommended system exhibits robustness and adaptability, as it features

a neural network at its core, and a complete data preprocessing pipeline to address any data inconsistencies.

Results produced by the application of evaluation metrics have numerically, on a scale, representing the performance and efficacy of the proposed methodology. The threshold sensitivity analysis revealed trade-offs between precision and recall and showcased visually the stability and confidence of the given model, as well as its priorities (which elements the model focuses on, such as “True Positives”, “False Positives” etc.). Threshold sensitivity analysis proves that the model prioritizes minimizing false malign instances, which is the optimal and expected model behaviour.

Recall, precision, and F1 score prove to be of critical importance in revealing and defining the model’s performance. Those metrics suggest that the model is robust, precise, accurate, and stable when dealing with unseen data.

As network systems continue to evolve, so do the cyber threats. Such threats may hide in plain sight, mimicking the normal data network flow, making them camouflaged to most network intrusion systems. Given approach suggests providing said systems with adaptability and artificial intelligence, by incorporating, as mentioned, a neural network at their core. In general networks, a larger flow of data is inevitable and can be put to use for neural network tuning and learning.

This study contributes to the ongoing evolution of proactive and adaptive defence mechanisms against cyber threats in networks. By leveraging neural networks and advanced data analytics, those defence mechanisms prove to be an intelligent adversary to cyber threats. This integration of neural networks offers a promising solution for further fortification of digital defences and guarding against ever-evolving threats.

Acknowledgements

The organizer gratefully acknowledges the work done by ALFA BK University ALFATECH 2024 International Scientific Conference for efforts done for the success of this event.

References

[1] Alom, M. Zahangir et al., A State-of-the-Art Survey on Deep Learning Theory and Architectures, *Electronics 8*, (2019.), DOI: 10.3390/electronics8030292.
[2] LeCun, Yann and Bengio, Y. and Hinton, Geoffrey, Deep Learning, *Nature 521*, (2015.), DOI: 10.1038/nature14539.
[3] Sarhan, Mohanad and Layeghy, Siamak and Portmann, Marius, NF-ToN-IoT, *The University of Queensland*, (2023.), (Dataset) DOI: 10.48610/2fa2ed6
[4] Scikit-learn, sklearn.preprocessing.MinMaxScaler, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
[5] TensorFlow, Classification on imbalanced data, https://www.tensorflow.org/tutorials/structured_data/imbalanced_data
[6] Ioffe, Sergey and Szegedy, Christian, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *arXiv preprint arXiv:1502.0367*, (2015.)
[7] Agarap, F., Abien, Deep Learning using Rectified Linear Units (ReLU), *arXiv preprint arXiv:1803.08375*, (2019.)
[8] Kingma, P., Diederik and Ba, Jimmy, Adam: A Method for Stochastic Optimization, *arXiv preprint arXiv:1412.6968*, (2017.)
[9] Sokolova, Marina and Lepalme, Guy, A systematic analysis of performance measures for classification tasks, *Information Processing & Management 45*, (2009.), DOI: 10.1016/j.ipm.2009.03.002.