Date of publication DEC-31, 2025, date of current version DEC-11, 2025.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.24.4.4328

Advanced Technique for Imbalance Mitigation in Predictive Monitoring and Anomaly Detection System

ANDRIY LUTSIUK[1], OREST LAVRIV[1], MYKOLA BESHLEY[2], MYKOLA BRYCH[2]
[1]Department of Electronics and Information Technology, Lviv Polytechnic National University, Bandera Str. 12, Lviv 79013, Ukraine
[2]Department of Information and Communication Technologies, Lviv Polytechnic National University, Bandera Str. 12, Lviv 79013, Ukraine

Corresponding author: Mykola Beshley (e-mail: mykola.i.beshlei@lpnu.ua)

ABSTRACT This paper presents an advanced approach to improving network traffic monitoring systems using machine learning algorithms. The main attention is paid to the problems of data imbalance and insufficient labeling in real communication systems. These problems often lead to inaccurate anomaly detection and unreliable system performance. To solve these problems, the paper proposes a dynamic class weighting technique that improves anomaly detection, especially when dealing with uncertain or unevenly represented data. The technique ensures that minority classes, such as malicious or anomalous traffic, are properly accounted for during model training, which improves overall detection accuracy. This approach provides the ability to dynamically change class weights based on new input data, and the simplicity of the model, because it is linear and does not have many layers, allows for relatively quick retraining. In addition, the paper describes an optimized data preparation process that facilitates efficient training of neural networks. These networks are integrated into proactive monitoring modules, which allows for real-time detection of network anomalies and potential threats. Although the proposed multiclass approach yields slightly lower global metrics (Precision 0.88, Recall 0.88) than the binary baseline, it significantly improves malicious traffic detection by introducing an additional class for uncertain samples, thus offering a more realistic and robust representation of network behavior. This proactive approach is particularly useful in today's communications environments, which are characterized by increasing traffic volumes and greater data diversity. By providing rapid detection and response to network breaches, the proposed solution increases the reliability and stability of networks, providing more robust protection against new cyber threats. The approach is particularly well suited for dynamic and complex networks, where traditional static monitoring methods often prove insufficient. The techniques presented in this article thus contribute to the development of more intelligent and responsive network monitoring systems that can cope with the complexities of modern communication infrastructures, where the demand for real-time analysis and anomaly detection continues to grow.

KEYWORDS network monitoring; proactive monitoring; communication systems; machine learning; neural network; anomaly detection; malicious.

I. INTRODUCTION

With the development of the Internet and the increasing number of connected devices, network infrastructure management and monitoring have become key areas of research and investment [1].

The constant growth of network traffic and the increasing number of devices, particularly within the Internet of Things (IoT), poses a significant challenge to researchers in terms of efficient monitoring, data management, and anomaly detection [2-5]. Although individual IoT devices are relatively simple, their large-scale interaction generates large amounts of heterogeneous data, which poses significant challenges for monitoring and ensuring network reliability [6].

In addition, the rapid development of mobile and wireless communications has led to the transformation of industries such as transportation and healthcare, which in turn has given rise to new cybersecurity threats [7]. Growing volumes of data and connected devices increase the vulnerability of systems to attacks. Traditional security methods can no longer effectively counteract sophisticated cyberattacks, such as hacking wireless sensor networks and data leaks [8].

In pursuit of goals, such as a high level of Quality of Service (QoS), all stakeholders are trying to somehow control and understand what is happening in the flow of data moving in communication networks [9]. Monitoring systems have become the tools that help to exercise this type of control.

Having a modular system, monitoring and control systems have become excellent assistants in understanding the state of the network, responding to events within the network, alerting and logging. However, as mentioned earlier, the amount of traffic is growing and monitoring systems are facing new challenges in the form of high traffic intensity and its diversity.

Today, researchers can solve this problem by improving proactive monitoring systems that allow to predict and prevent the occurrence of an atypical situation in the network in advance. The method of improvement is the use of machine learning algorithms in the modules of these systems [10].

There are already studies where the use of machine learning algorithms of various types shows an accuracy of classification of incoming traffic of more than 90% [11], which is a strong evidence in favor of the feasibility of this approach. However, such systems are often run on preprepared data, which in their structure contain quite a few examples of both normal and malicious traffic, which somewhat simplifies the learning process and ultimately shows very good results. However, in real systems, there are problems of imbalance, poor labeling, and heterogeneity, which introduces new complexities and challenges to the process of training and testing models in communication network monitoring systems [12].

To mitigate these problems, optimal training sample selection and hybridization of neural networks with fuzzy logic are considered promising directions to improve anomaly detection accuracy in imbalanced network environments.

The paper presents a method to overcome the problems of imbalance and limited data, which are critical for the functioning of proactive monitoring systems based on machine learning algorithms. An improved approach to data preparation and its further use for training neural networks in the context of proactive monitoring systems is proposed. The process of developing a neural network that is integrated into the module for predicting and detecting anomalies in network traffic, including malicious actions, is described.

The paper is structured as follows. Section 2 provides a comprehensive review of existing research in this field, identifying key challenges and issues. Section 3 describes the architecture and functional components of network monitoring systems. Section 4 provides a detailed description of the proposed imbalance mitigation technique for predictive monitoring systems. Section 5 presents the results and analysis, followed by a discussion in Section 6. Finally, Section 7 contains concluding remarks and outlines future challenges.

II. RELATED WORKS

The idea of using machine learning in monitoring systems to predict the future state of a communication network is not new. There are many studies that have proven the effectiveness of machine learning algorithms in prediction in one way or another. In particular, study [13] shows that some solutions based on machine learning algorithms are close to 90% and sometimes 97% accurate. The study [14] shows accuracy rates of 93% and higher, which is a good result.

However, despite numerous developments, researchers continue to face a number of challenges and problems related to the specifics of telecommunications systems.

The first problem is data sets that do not correspond to the actual situation in real systems. Due to the peculiarities of data collection in telecommunication systems, researchers are severely limited in the available data sets, especially when it comes to labeling a malicious flow in sufficient quantity. Therefore, in their works [15-18], the authors use publicly available datasets where there are clearly defined two classes describing abnormal and normal traffic. The number of records in such datasets is balanced to ensure equal representation of both states. Also, analyzing the datasets presented in studies [19-21], it is clear that these can even be synthetically created data, where malicious traffic is modeled rather than natural, which does not quite reflect the true picture of heterogeneous flows in real systems.

Although features of this type are less common, researchers use the accuracy metric as an indicator of the final result [22]. In fact, for idealized datasets where there is class parity, this can work and really reflect the true state of affairs. However, when classes are unbalanced, the problem arises that the accuracy rate of 90% is only an indicator of the ideal classification of the class that prevails in this set. In addition to accuracy, MAE metrics can be used [23]. In the context of binary or multiclass classification, MAE does not take into account the probability of predictions, but only the absolute differences between predicted and actual values. Nevertheless, the use of these metrics is still a consequence of the datasets that allow this and show good results in the end. But, as noted earlier, in real telecommunication systems, there is heterogeneity, insufficient or poor labeling of real data, and insufficient number of records of certain classes, which together does not allow using accuracy as an indicator.

The issue of data imbalance, where the volume of regular instances significantly exceeds that of anomalies, poses substantial challenges, especially for traditional classification models. This imbalance, commonly referred to as the class imbalance problem, can severely hinder the model's ability to accurately detect rare but critical anomalies. Authors [24] applied reinforcement learning along with SMOTE to enhance the performance of one classifier on NSL-KDD dataset whereby data is imbalanced. There are numerous other experiments comparing performance metrics of SMOTE, ROS, Near-Miss1, and Near-Miss2 methods and managed to get up to 82% peak accuracy together with an F1 score of 82.4%. Authors [25] employed preprocessing via one-side selection and SMOTE sampling in order to solve data imbalance issue. While performing their work, the authors obtained 83.58% and 77.16% accuracy using the hybrid convolutional neural network and bidirectional long short-term memory model on the NSLKDD and UNSW-NB15 datasets respectively. In the same manner, authors [26] presented a unified model employing LSTM cornered by chaotic butterfly optimizer along with particle swarm optimization for bettering intrusion detection performance. They managed to achieve 93.09% and 86.89% respective accuracy on KDDTest+ binary dataset and KDDTest-21 dataset. Authors [27] dealt with the problem of network anomalies detection and proposed deep learning model with attention, CNN and elements of integration providing a solution. While Jony and Arnob [28] provided a useful comparative evaluation of machine learning algorithms on the CIC-IoT2023 dataset, their study did not address the issue of data imbalance.

According to the literature review, the main problem of existing monitoring systems is the insufficient amount and labeling of data, which makes it difficult to use binary models to accurately detect anomalies. To solve this problem, it is necessary to develop a new proactive monitoring technique capable of working with incompletely labeled data. The key elements of this technique should include: active learning to automate labeling, hybrid models for more accurate traffic analysis, dynamic adjustment of class weights to balance data, and early warning modules for timely response to threats.

III. ARCHITECTURE AND FUNCTIONAL COMPONENTS OF NETWORK MONITORING SYSTEM

A monitoring system is a complex of software and hardware tools designed to detect network attacks, diagnose malfunctions and other problems that arise in the system in order to increase its fault tolerance and ensure a high level of user service.

Monitoring systems cannot be considered as simple, monolithic solutions. As a rule, such systems consist of several functional modules, the number and composition of which varies depending on the specific tasks assigned to the system and the type of traffic to be analyzed. Considering a simple example of a monitoring system, the following components are primarily distinguished:

- Monitoring agent refers to a unit responsible for collecting traffic data from various systems for further processing;
- Main monitoring server is a server where, if necessary, preliminary data processing takes place, previous metrics are stored, information is analyzed, and decisions regarding alerts are made;
- Alerts system is a notification system that informs the relevant individuals or other network nodes about potential changes in the state of the telecommunications network (it can be implemented as part of the main monitoring server).

Monitoring systems, in turn, are generally divided into active and passive types based on their operation mode [28].

Active monitoring methods are often referred to as "synthetic" because this approach does not use actual user data. Instead, the tools employed in this monitoring aim to predict the potential performance of the network by simulating its current behavior. Active monitoring seeks to provide a comprehensive view of network performance in real time. Additionally, it allows for measuring network performance through various metrics and key indicators, including latency, response time, jitter, packet loss, etc.

Passive monitoring is based on analyzing actual user data within the network. Active monitoring generates small, regular data packets while operating, whereas passive monitoring uses real, holistic, and significantly larger data sets, providing a more accurate snapshot of the network's current state.

Each of these approaches has its advantages and disadvantages, which can be summarized as follows:

- Active monitoring enables the identification of potential problems before they occur, but it requires more resources, and its accuracy is based on predictions;
- Conversely, passive monitoring provides a comprehensive understanding of overall network performance at the current moment, as it uses actual network data. However, any issue identified by this method is an existing problem that requires immediate resolution.

In essence, one method generates additional data and uses the network while trying to predict future changes in the system's state, whereas the other method uses real user data to describe the current system state.

Traditionally, passive monitoring systems have gained wider adoption and were primarily focused on addressing issues after they had been detected. This approach to monitoring is referred to as reactive monitoring. A generalized example of the operation of a reactive system is shown in Fig. 1.

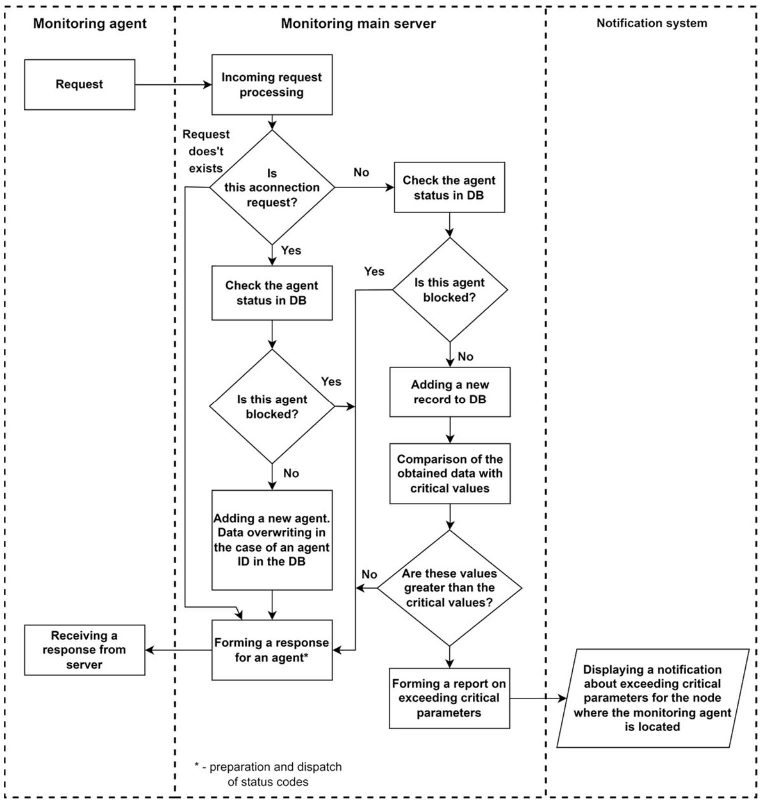

Monitoring agent
Monitoring main server
Notification system
Request
Incoming request processing
Request does't exists
Is this aconnection request?
No
Check the agent status in DB
Yes
Check the agent status in DB
Yes
Is this agent blocked?
No
Is this agent blocked?
Yes
Adding a new record to DB
Comparison of the obtained data with critical values
No
Adding a new agent. Data overwriting in the case of an agent ID in the DB
No
Are these values greater than the critical values?
Receiving a response from server
Forming a response for an agent*
Forming a report on exceeding critical parameters
Displaying a notification about exceeding critical parameters for the node where the monitoring agent is located
* - preparation and dispatch of status codes

Figure 1. Block diagram of a reactive monitoring system using an agent.

In this figure, three components of the system are presented an agent that provides information from network nodes about the observed parameters, a monitoring server that processes the received information and makes decisions based on it, and a notification module responsible for delivering status messages about the system in real-time, if necessary.

In monitoring systems of this type, predefined response and behavior scenarios are available for possible failures. However, when it comes to applying these predefined scenarios, by the time the system status change is detected, the critical response time may have already passed, potentially leading to a negative impact on user experience and service quality, while the system continues to experience failures. To better understand this concept, consider a situation where an incident involving critical server load occurs. When the critical threshold is reached, the system sends a notification to another module, which, in turn, triggers the activation of an additional server to balance the load. Here arises a dilemma between activating the server too early, thus incurring unnecessary financial costs, and activating it too late, where a certain number of users may have already experienced slow response times or even errors.

However, such a situation could have been anticipated at the traffic growth stage by analyzing the number of connections and other metrics, allowing for the timely activation of the additional server. This is where predictive monitoring comes into play, which aims to foresee system state changes. Predictive monitoring is an evolutionary step beyond reactive monitoring. This type of monitoring is focused on alerting about the existence of a potential problem before it escalates to the point where reactive monitoring would respond. An example of such a system is depicted in Fig. 2.

The first key feature distinguishing predictive systems from the previously discussed systems is the presence of an additional module, known as the prediction module. The primary goal of this module is to analyze incoming traffic and predict system state changes in advance.

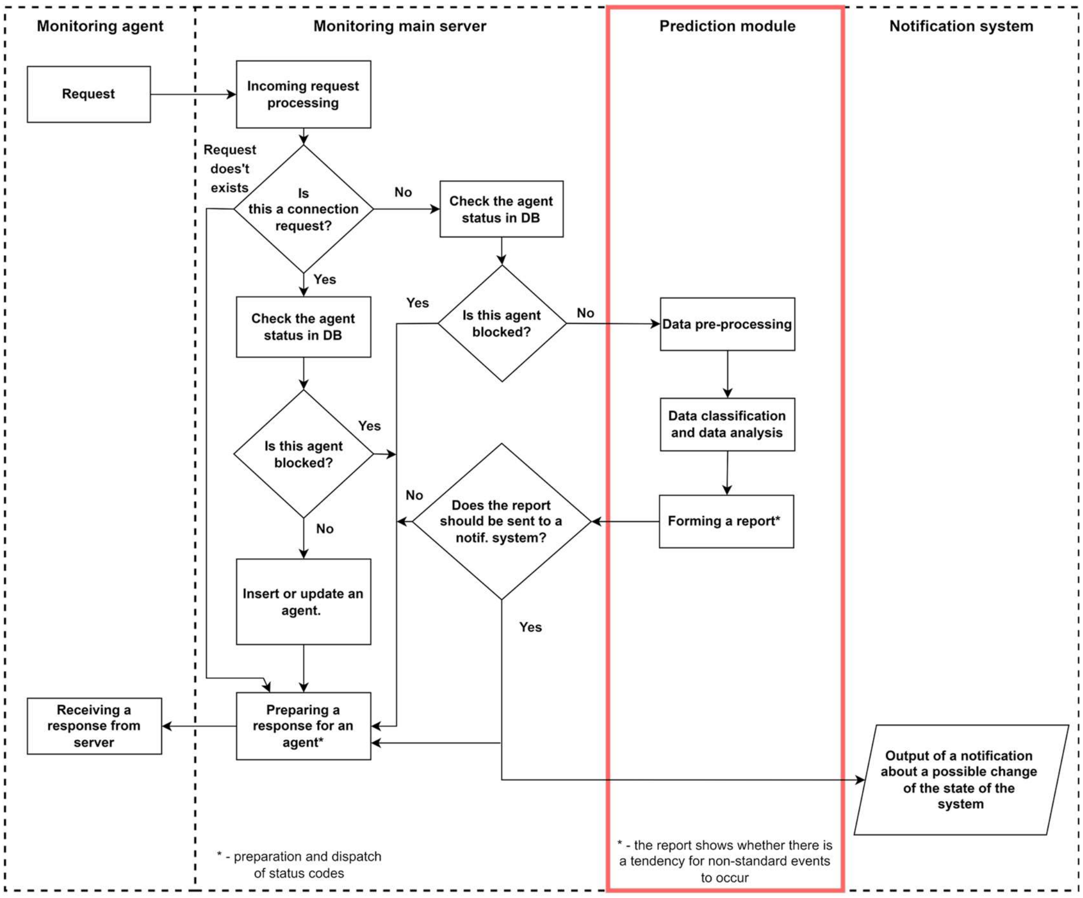

Monitoring agent
Monitoring main server
Prediction module
Notification system
Request
Incoming request processing
Request does't exists
Is this a connection request?
No
Check the agent status in DB
Yes
Check the agent status in DB
Yes
Is this agent blocked?
No
Data pre-processing
Data classification and data analysis
Yes
Is this agent blocked?
No
Does the report should be sent to a notif. system?
Forming a report*
No
Insert or update an agent.
Yes
Receiving a response from server
Preparing a response for an agent*
Output of a notification about a possible change of the state of the system
* - preparation and dispatch of status codes
* - the report shows whether there is a tendency for non-standard events to occur

Figure 2. Block diagram of a predictive monitoring system using an agent.

A real-time predictive monitoring system collects, processes, and analyzes data from sensors, hardware and software solutions, and specific application or operating system logs, and provides an estimate of the probability of a change in the system's state. In general, this can be described as a trend assessment. A trend means a change in the monitoring data indicators in the future and how this may affect the system as a whole.

Predictive monitoring aims to avoid the time delay that occurs between an event and the reaction to the event. In the previously discussed, reactive approach to monitoring, there is a certain time delay between the moment an event occurs T_{event} and the moment the event is reacted to T_{react}. This time delay exists because of alerting, because first there is a notification T_{notif}, and only after that comes the reaction. In general, this time delay can be described by the following formula:

$$\Delta T = T_{notif} - T_{event}. \quad (1)$$

In a reactive monitoring system, the time difference between the occurrence of an event and the system's response to the event will always be a positive number. This is due to the peculiarity of the system's operation, since it is triggered only when an event occurs, and does not anticipate it.

In fact, the new component of the system highlighted in Fig. 2 allows us to overcome the time delay. This module analyzes the input data and tries to predict the change in the system's state right then and there. This component introduces a new prediction T_{pred}. The prediction occurs earlier, before the event itself T_{event}. Thus, the reaction time to an event in a proactive system can be described by formula:

$$\Delta T = T_{pred} - T_{event}. \qquad (2)$$

In this formula, ΔT is the time that shows us the time delay from predicting an event to its occurrence. Since the predictive monitoring system works proactively, we have a situation where, in the case of a correct prediction, the time delay will be negative, which means gaining time to react to the event before it occurs. In fact, as for the event itself, the time of its occurrence is a value predicted with a certain accuracy that must be anticipated in order to avoid it, since the event may already be followed by a change in the state of the entire system.

The "system state" refers to the set of current parameters and metrics that reflect the functional condition of a specific system at a given point in time. Such parameters may include network load, signal quality, data transmission speed, and other key indicators that directly impact the overall performance of the system. Assessing the system state not only enables analysis of its efficiency but also helps in detecting anomalies and predicting potential failures or malfunctions. Mathematically, the system state can be described as a set of parameter values and their corresponding coefficients, as shown in formula 3:

$$S_{system} = k_1 \cdot V_1 + k_2 \cdot V_2 + \cdots + \ k_n \cdot V_n. \qquad (3)$$

In formula 3, the symbol V is the percentage value of the utilization of one of the system components at a given time. In turn, the symbol k is the coefficient of such a system component, i.e. its importance relative to other components in determining the system state per unit of time. Individually, the sum of the coefficients k must always be equal to 1. As a result, the value of S_{system} ranges from 0 to 100. Thus, the sum of system resource utilization and their coefficients is an indicator of the system state.

Taking into account Formula 3, only systemically important parameters can and should be selected from the list of system parameters available for tracking. In turn, parameters that have a low impact on the system state can be ignored to simplify calculations.

The simplest example of system states, in the context of communication systems and networks, are:

- *normal state* is a mode of operation in which the system is in its own or a certain normal (regular) mode and has no tendency to change it;
- *critical state* is a mode of operation when the system experiences inefficiency, malfunction, and even complete failure.

Usually, during the initial analysis of indicators, it is not possible to obtain all the indicators that can become indicators of the system's state, and often they are obtained already in the process of testing algorithms on the data obtained. Nevertheless, the analysis should take place in advance and the parameters should be at least partially determined. Observing the defined parameters allows us to understand the state of the system in this case.

Summarizing the concept of state, it is understood that only systemically important parameters are considered to describe the state in telecommunication systems, the simplest example of which is CPU, RAM, network usage, ROM. Also, such concepts as “normal system state” or “critical system state” should be defined in advance, because for each individual system the concept of, for example, normal state may differ.

IV. IMBALANCE MITIGATION FOR PREDICTIVE MONITORING SYSTEM

Therefore, we have described the concepts of system state, prediction, and the domain in which the study is conducted. Now, directly in the context of the article, the object of study is the prediction module, so the focus is on this part of the overall prediction system. The prediction module is a part of the monitoring system that receives data from the network flow, analyzes each record or action made in this flow, and predicts the further state of the system. In fact, the prediction module deals with classical classification tasks, but with the difference that it is primarily interested in identifying a specific class as accurately as possible. In the context of this article, a class is nothing more than the defined nature of a user request to the system under study.

Based on the previously discussed points, the issue of class imbalance arises. Class imbalance is a situation in machine learning where the number of examples in one class significantly exceeds the number of examples in other classes within a dataset, which can lead to a decline in the model's classification performance. The situation is further complicated by the fact that, under normal operating conditions, the monitoring system predominantly processes normal user traffic, making class parity for model training almost impossible. Malicious traffic occurs much less frequently. Additionally, there is another class of traffic often referred to as outliers or undefined traffic. Although it may sound unusual, there is a significant amount of traffic that cannot be clearly labeled as either allowed or forbidden, and this type of traffic also needs to be addressed.

Understanding the nature of the problem and why it arises, Fig. 3 illustrates the proposed changes to the data processing and model training workflow.

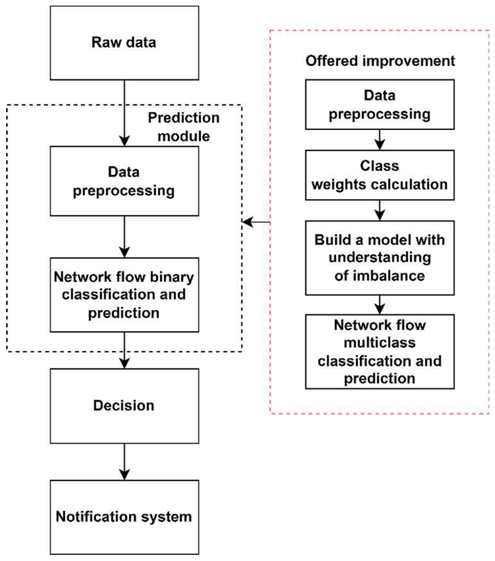

Raw data
Prediction module
Data preprocessing
Network flow binary classification and prediction
Decision
Notification system
Offered improvement
Data preprocessing
Class weights calculation
Build a model with understanding of imbalance
Network flow multiclass classification and prediction

Figure 3. Simplified view of the predictive module with proposed modifications.

The first proposal is to adjust the approach to data analysis and preparation. At this stage, it is crucial to assess the severity of the class imbalance and calculate the imbalance ratios to determine appropriate class weights.

The second proposal is to train the model with the understanding that the dataset is imbalanced, meaning that the traditional accuracy metric is not suitable in this case. More appropriate metrics exist that do not ignore less frequent classes and focus on better identifying the desired class.

To address the issue of traffic classification in communications systems, we employed an approach based on using high-quality data for model training. One of the datasets we used is the LUFlow Network Intrusion Detection Data Set [29]. This is a flowing dataset specifically collected and labeled for training models to detect network intrusions. The data was gathered using honeypots, which are systems designed to act as decoys for malicious traffic (Fig. 4). This allowed us to collect valuable information for further analysis and threat detection. Honeypots are essentially resources or devices used to attract malicious traffic [30]. The primary purpose of honeypots in the system is to intentionally expose the network to attacks or unauthorized probing, enabling the collection of information that can later be analyzed. To capture telemetry, the Cisco Joy tool was used within such a node of the communications system. Joy is a BSD-licensed software package based on libpcap for extracting data features from live network traffic or packet capture files (pcap).

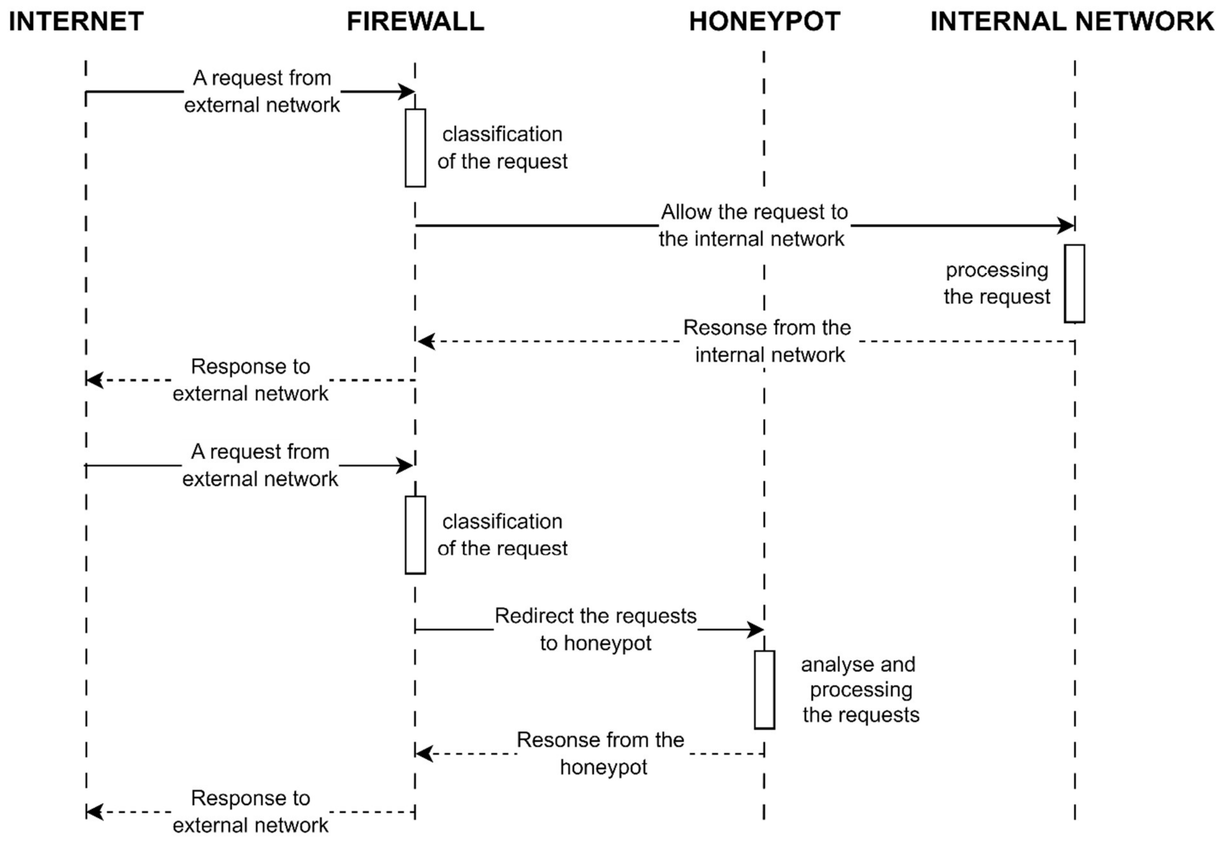

INTERNET
FIREWALL
HONEYPOT
INTERNAL NETWORK
A request from external network
classification of the request
Allow the request to the internal network
processing the request
Resonse from the internal network
Response to external network
A request from external network
classification of the request
Redirect the requests to honeypot
analyse and processing the requests
Resonse from the honeypot
Response to external network

Figure 4. Simplified example of the honeypot algorithm.

It is important to note that the dataset includes traffic flows that could not be classified as malicious but are also not part of the typical telemetry profile. These data points are labeled as outliers, included to encourage further analysis to uncover the true intent behind their actions.

Given the system in which the research is being conducted, it is advisable to consider in detail the dataset itself that will be used (Table 1) and analyze examples of records that characterize the defined classes (Table 2).

Referring to the specific examples shown in Table 1, we can immediately notice that the so-called malicious traffic example, which will further interest us in the study, has a sharp difference in the absence of source port and destination port (explanation of each parameter is described in the following paragraphs).

Table 1. Dataset values

Field name	Definition
src_ip	anonymized source ip address
src_port	source port number
dest_ip	anonymized destination ip address
dest_port	destination port number
protocol	protocol number under which the flow works
bytes_in	number of bytes transmitted from the source
bytes_out	number of bytes transmitted from the destination
num_pkts_in	number of data packets from source to destination
Entropy	entropy in bits per byte of data fields in the flow
total_entropy	total entropy in bytes for all bytes in the data fields of the flow
duration	time of the flow duration to the nearest microsecond
label	definition of the flow. markup.

Table 2. Used traffic examples

Field name / Labels	benign	malicious	outlier
src_ip	786	786	786
src_port	68		47613
dest_ip	786	786	786
dest_port	67		31306
protocol	17	1	6
bytes_in	0	8	0
bytes_out	600	8	0
num_pkts_in	0	1	0
entropy	2	1	1
total_entropy	1.615865	2.75	0.0
duration	969.5192	43.99	0.0
label	7.17	8.4e-5	0.0

However, this is just one example and it is not a general indicator. In any case, for a better understanding of the data set, it is worth referring to the pairwise correlation using the Pearson method and analyzing the dependence of the parameters (Fig. 5).

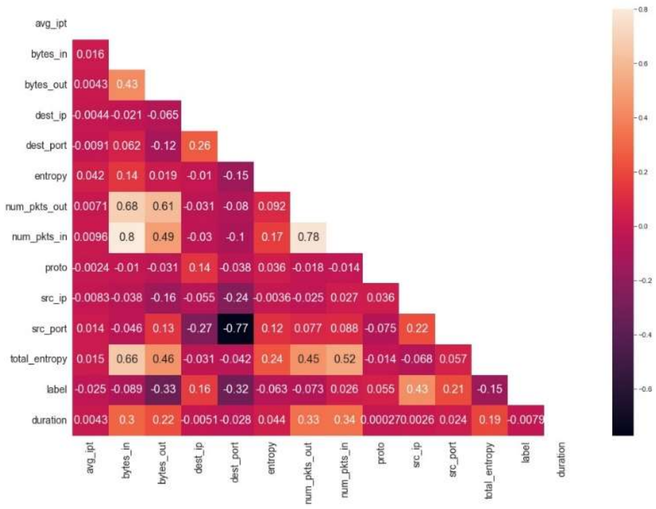


Figure 5. Pairwise correlation by Pearson

Based on the obtained pairwise correlation values, the label value is strongly correlated with such parameters as: bytes out, destination port, destination ip, source port, source IP, total entropy. However, other parameters should not be excluded from the analysis because they provide a combination that, in general, describes the state of the system at a particular time.

Another important factor to be determined is the imbalance indicator, which was mentioned earlier. To do this, it is needed to determine the number of available records for each of the known classes.

Table 3. Number of records for each class

Class name	Count	Percent
All	3826947	-
Benign	2162576	56.51%
Malicious	910391	23.78%
Outlier	753980	19.71%

If we look at Table 3, which counts the number of all classes and each class individually, we can see that the malicious class that will need to be identified is only about a quarter of the total dataset. According to the imbalance classification provided by Google in their training programs [31], the imbalance of the class is moderately light, as it is in the range of 20 to 40%. However, the gap of 20-40% is still quite large, and taking into account the number of records, we can say that the figure of 23.78% for this class is an indicator of moderate imbalance.

To ensure a representative and unbiased dataset for training and evaluation, the data were sampled from a large continuous data stream collected over several days. The selected subset covers typical operating conditions and includes both common and less frequent events, ensuring variability in the input features and stability of model training.

After data cleaning and preprocessing, the dataset was divided into training and testing subsets using a standard randomized partitioning approach. Specifically, the train_test_split function from the scikit-learn library. This function randomly assigns 80% of the samples to the training set and 20% to the testing set, while fixing the random seed (random_state=30) to ensure reproducibility of the results.

V. RESULTS AND ANALYSIS

According to Table 2, the label field is a text field and, accordingly, the name of each class is also written in text form. Such a format for presenting markups is not suitable for further research and should be converted to a numerical format. From this point on, a situation arises when the further execution of the task differs in approach due to the peculiarities of the task interpretation. In the case of the data set under study, there are clearly 3 classes, which is nothing more than a multi-class classification task.

However, there is another approach when the goal is to identify malicious traffic and consider everything else as normal traffic. In this case, the type of problem changes from multi-class to binary. To better understand the problems of each of the tasks, we need to consider their final results.

A. BINARY CLASSIFICATION IN UNBALANCED DATA SETS

Binary classification is a type of machine learning problem where the model is trained to distinguish between two different classes, i.e., to determine which of the two possible options each input example belongs to. In the case of the dataset under study, if we follow the commonly used path and ignore the outlier class, as it represents outliers of unlabeled data, we will be left with only two classes that are subject to binary classification.

Since there are fewer classes, the outlier data is ignored, and the imbalance of the classes has changed slightly. From Table 4, we can conclude that benign has increased its dominance, and malicious, in turn, has begun to occupy a third of the dataset. To fit the data for this case, it is possible to use the bincount function of the numpy library, as shown in the examples for classifying unbalanced classes from Keras [32]. The results of weight selection are shown in Table 5.

Table 4. Number of records for each class

Class name	Count	Percent
All	3072967	-
Benign	2162576	70.37%
Malicious	910391	29.63%

Table 5. Defined class weights

Class name	Weight
Benign	2.3118e-06
Malicious	5.6707e-06

Knowing the type of problem, features of the data set, and weighting factors, we can proceed to developing the model. Developing a model is an iterative process that requires constant experimentation and tuning. The layers of the model, the number of neurons, are usually selected in practice through trial and error, and are not determined on the first try. After a certain number of retraining sessions, the network structure looked like the one shown in Fig. 6. The Keras tool, in particular its Sequential class, is used to build the model.

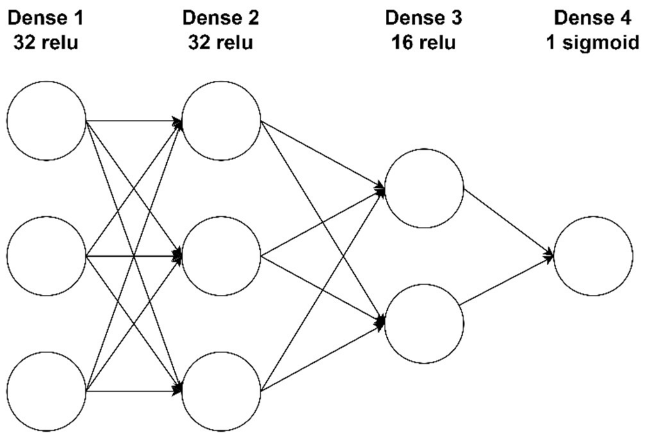
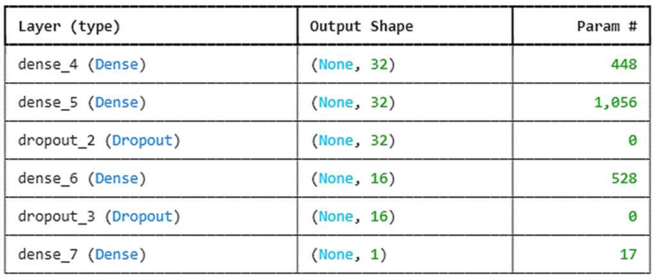


Figure 6. A neural network model for the binary classification task.

In Fig. 7, we show the number of parameters for each layer and mention dropout layers. The dropout layer is a regularization technique in neural networks that randomly disables some neurons during training to prevent overfitting and improve the generalization ability of the model.



Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 32)	448
dense_5 (Dense)	(None, 32)	1,056
dropout_2 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 16)	528
dropout_3 (Dropout)	(None, 16)	0
dense_7 (Dense)	(None, 1)	17

Figure 7. Number of parameters in the layers of the neural network.

For the loss function, *binary_crossentropy* is used, which is a classic approach in such tasks. As for the metrics, in the task of detecting malicious traffic in a telecommunications network, as a rule, more importance is given to the recall indicator. This is due to the fact that malicious traffic can cause significant damage to the network and its users, and therefore it is very important to detect as many malicious packets as possible. However, a high recall metric can lead to an increase in the number of false positives (i.e., the number of packets that are incorrectly classified as malicious).

This can lead to unnecessary actions on the part of the security system, such as blocking legitimate users or slow connections. In turn, if we focus on the accuracy metric, we can find that high accuracy is achieved due to the imbalance of classes and the constant, correct prediction of non-malicious traffic that is not the ultimate goal. Based on the results presented in Table 6, the model performed almost perfectly during training.

Table 6. Results of the last epoch of model training

Class name	Count
False negative	430
False positive	547
True negative	1729478
True positive	705109
Precision	0.9992
Recall	0.9994

Based on this data alone, the model was wrong only 977 times out of more than 2 million for the task at hand. On the test dataset, which contains 432547 records for Benign and 176344 for Malicious, the results are expected to be perfect prediction of malicious and normal traffic (Fig. 8).

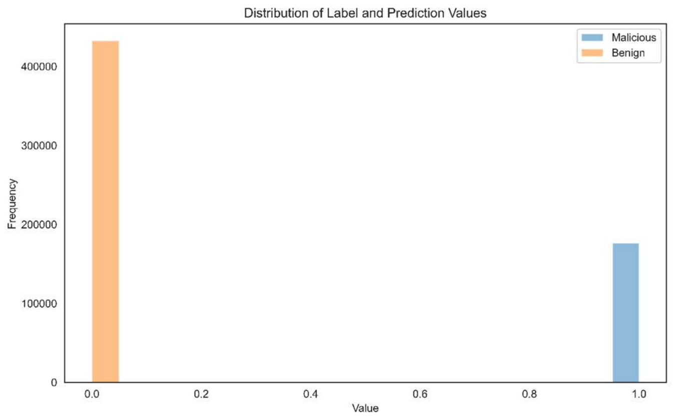


Figure 8. Traffic forecasting.

This could be the end of the experiment, but there is a problem that these are only the results for perfectly labeled data. In real systems, unfortunately, there is no such well-labeled data and no perfect separation into two classes. Therefore, working with outliers is a necessity.

B. MULTI-CLASS CLASSIFICATION IN IMBALANCED DATASETS

Multi-class classification in machine learning is a type of task where a model is trained to distinguish between more than two classes by determining which of several possible options each input example belongs to [33-37]. In the case of the dataset under study, in the previous step, one of the classes was ignored and binary classification was performed, and in this step, the outlier class is also investigated.

As with the binary classification, it is necessary to calculate the weights of the classes for further use in model training. In this case, we used a slightly different approach to calculating class weights and used the *compute_class_weight* function of the scikit-learn library. The results of calculating the weights are presented in Table 7.

Table 7. Defined class weights

Class name	Weight
Benign	0.5832
Malicious	1.4298
Outlier	1.7066

Knowing the type of task, features of the dataset, and weighting factors, we can proceed to building the model. The model development did not lead to any major changes in the architecture. Only the output layer that requires multi-class classification has changed.

In this layer, as shown in Fig. 9, the output neuron is no longer sigmoid, but softmax. The sigmoid activation function is used for binary classification and returns a value between 0 and 1, while the softmax activation function is used for multiclass classification and returns the probabilities for each class summed to 1.

Since the overall architecture of the neural network remains unchanged, including the number of layers and the size of hidden units, the training time of the model also remains virtually unchanged. After all, the main modifications only concern the weighting of classes and the activation function in the output neuron.

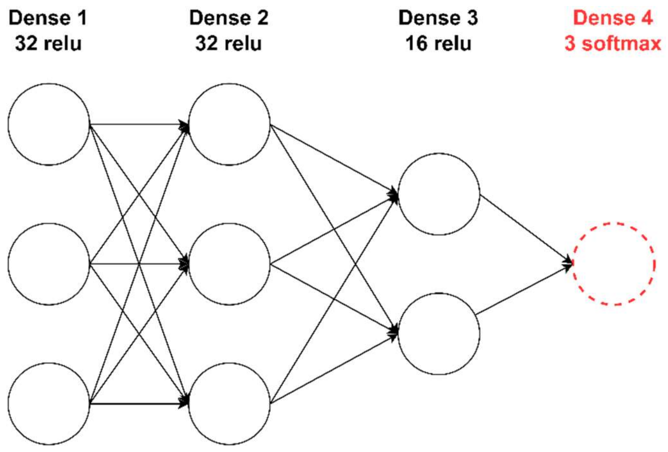
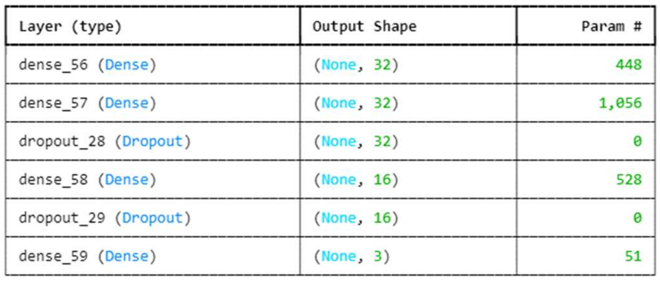


Figure 9. A neural network for the task of multi-class classification.

Fig. 10 shows the number of parameters for each layer and the dropout layer. Compared to the parameters in Fig. 7, the changes are almost invisible.



Layer (type)	Output Shape	Param #
dense_56 (Dense)	(None, 32)	448
dense_57 (Dense)	(None, 32)	1,056
dropout_28 (Dropout)	(None, 32)	0
dense_58 (Dense)	(None, 16)	528
dropout_29 (Dropout)	(None, 16)	0
dense_59 (Dense)	(None, 3)	51

Figure 10. A neural network for the task of multi-class classification.

Also, in this task, the loss function is changed from *binary_crossentropy* to categorical_crossentropy. For *categorical_crossentropy*, there is a certain peculiarity in its operation, namely that the function expects class labels to be passed in the *one_hot* format. The *one_hot* format is a way of representing class labels where each class is encoded by a vector that has a value of 1 at the position corresponding to that class and a value of 0 at all other positions. This allows the loss function to correctly calculate the difference between model predictions and true labels [32].

To obtain this format, the *to_categorical* function, which is part of the Keras package, was used on a dataset that described the label for each record in the dataset. So, in the process of training the model, the last epoch of training showed the results presented in Table 8.

Table 8. Results of the last epoch of model training

Class name	Count
False negative	361755
False positive	360617
True negative	5692039
True positive	2664573
Precision	0.8808
Recall	0.8805

At first glance, the results for misclassification of positive and negative records are enormous compared to the results in Table 6. However, it is worth remembering that out-liers are also used here, which account for almost 20% of the entire dataset. Therefore, it is worth reviewing the results of the predict function for the test dataset. The test dataset has 756582 records, which are described in detail in Table 9.

Table 9. Defined class weights

Class name	Count
Benign	432850
Malicious	176371
Outlier	147361

Considering the prediction of benign traffic, Fig. 11 shows how the prediction fully understands what a normal data flow is and classifies it accordingly. In fact, the number of correctly classified benign traffic and its number in the dataset are the same.

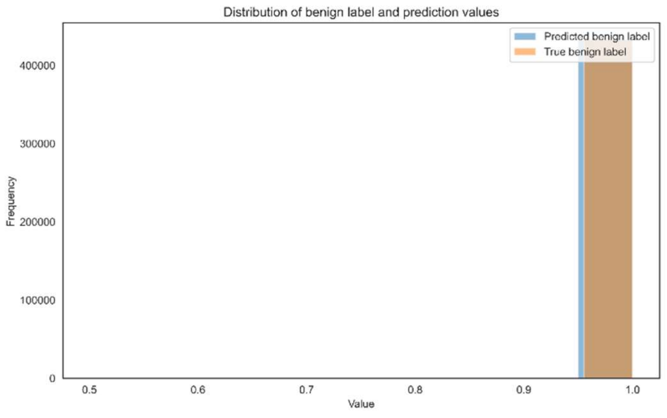


Figure 11. Benign flow prediction.

With the outlier data flow, the situation is already radically different (Fig.12). The model missed a lot of traffic of this type and did not classify it as any kind of outlier. The most sensitive interval is 0.8-0.85. This is where most of the errors were made.

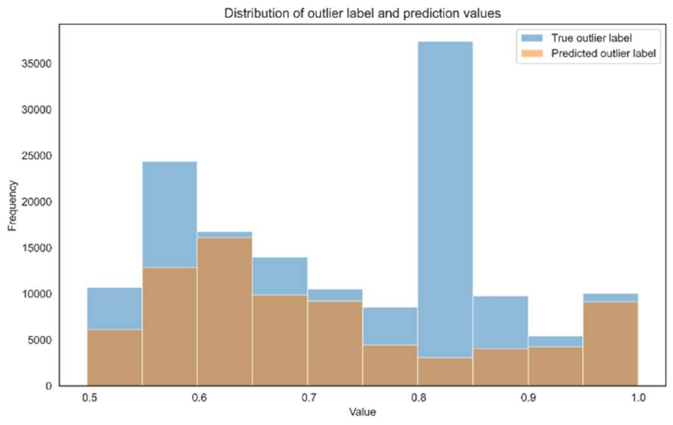


Figure 12. Outlier flow prediction.

As for the malicious flow, the situation is very similar to Figure 11, but in this case, the model has made more malicious traffic labels in the same range of 0.8-0.85 than there actually is.

In fact, if we compare Figures 11, 12, and 13, we wonder whether the outliers are the same overlap that the model shows for malicious traffic. To check this, we need to combine the graphs from Figures 12 and 13.

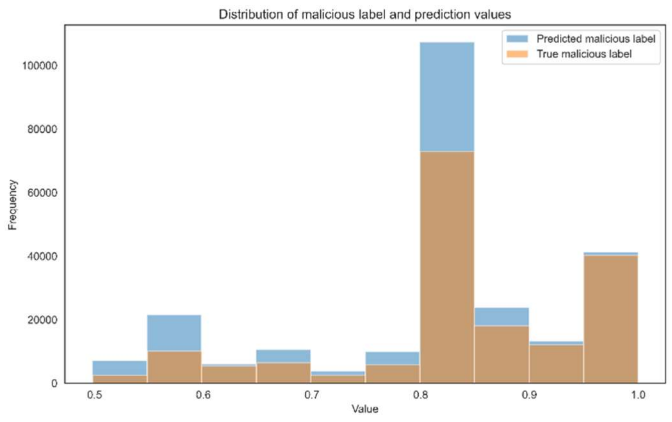


Figure 13. Distribution of malicious label and prediction values.

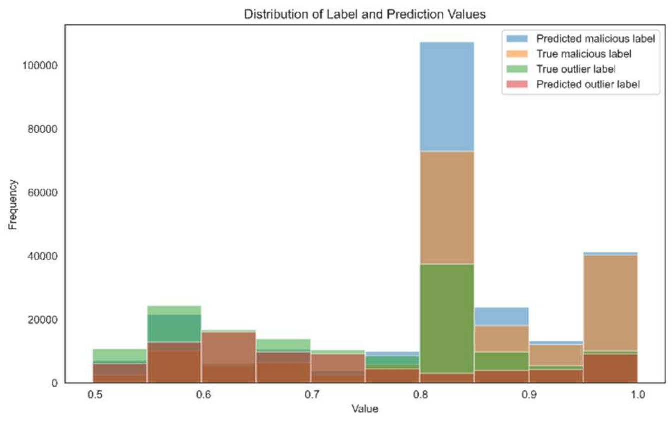


Figure 14. Distribution of label and prediction values.

These indicators should be considered in absolute terms. To do this, we need to supplement Table 9 into Table 10.

Table 10. Number of records for each class

Class name	Count	Predict	Difference
Benign	432850	432803	- 0.01 %
Malicious	176371	244790	38%
Outlier	147361	78989	- 53.6%

Based on the data in Table 10, the result of the model has several aspects. First of all, the QoS for a normal user should not suffer, since under normal operation and normal traffic, the misclassification of normal traffic is close to 0. As for emissions and malicious traffic, the situation is a bit more complicated. In fact, it has been possible to build a pessimistic system that can label most of the traffic that does not look like normal traffic as malicious. In general, this can be a problem in terms of using additional resources. On the other hand, the theoretical losses due to the passage of malicious traffic are not proportional and can cause much more damage. Comparison of existing labels and their coincidence with the prediction for the binary classification scenario is depicted in Fig. 15 and with the multi-class classification scenario in Fig. 16.

The results of the experiment show that due to the heterogeneity of data in communication systems and the disproportionate sets of labeled data representing each of the studied classes, the behavior of the trained model may differ from its settings. In a task where it is necessary to identify malicious traffic, outlier or unlabeled traffic, which exists in abundance in real systems, will also be captured and labeled as malicious.

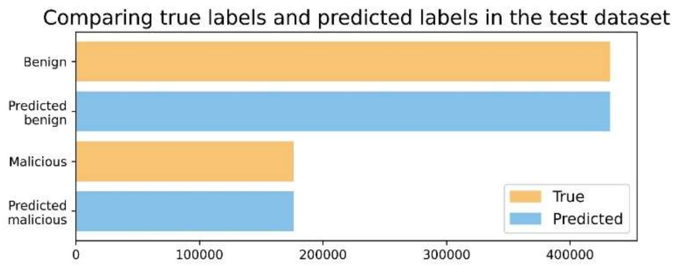


Figure 15. Comparison of existing labels and their coincidence with the prediction for the binary classification scenario.

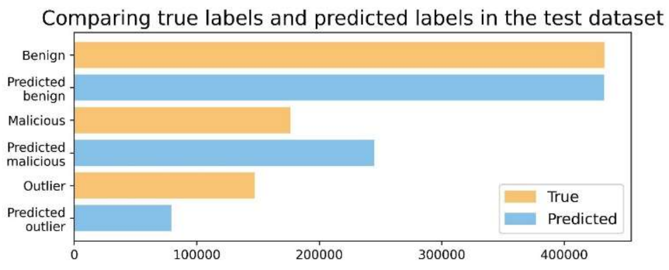


Figure 16. Comparison of existing labels and their coincidence with the prediction for the multi-class classification scenario.

Working with real data will always carry a certain percentage of errors. However, depending on the model settings and thresholds that will satisfy the conditions of the task, the percentage of false classifications will migrate from one class to another. Thus, the experiments once again confirmed the problem of unlabeled datasets.

VI. DISCUSSION

Finally, addressing the implementation of neural networks, there are many controversial issues and challenges that both businesses and researchers are working on. As research has shown once again, the most effective way to train neural networks is still training based on well-labeled data. However, since real telecommunications systems generate petabytes of unlabeled or poorly labeled data, only a comprehensive approach can resolve this issue.

Difficulties in using neural networks for structured data. Structured data refers to a standardized format that organizes data into tables with rows and columns, such as network traffic data. However, the earliest and most successful applications of machine learning report challenges with unstructured data, such as video, images, text, and audio. Some machine learning experts oppose the use of neural networks for structured data, as they believe that labeled structured datasets are not large enough to train machine learning algorithms. Furthermore, they argue that classical machine learning algorithms, such as KNN and SVM, are much simpler and more understandable, making them the only suitable options for use. This issue requires deeper investigation, as there is substantial evidence to the contrary.

Most machine learning algorithms are designed to be trained and utilized on devices with sufficient resources. Training neural networks with a large number of samples and parameters requires devices with substantial computational power, memory, and energy. This directly conflicts with the growing interest in deploying resource-constrained devices (e.g., IoT devices) equipped with AI technologies and AI-based applications.

Unlike other ML applications, network technologies suffer from high dynamism, thus requiring model retraining to adapt to new network situations.

Network heterogeneity issue stems from the fact that networks lack a unified theory that can be applied across all networks. This means that network behavior is heterogeneous, depending on factors such as different topologies, equipment, scale, and applications. This leads to an important point: machine learning models, for further use in prediction modules, must be trained for each network separately. This issue is also a crucial research direction, as the more flexible the system, the more potential telecommunications networks it can cover.

Despite the promising results, the proposed approach has certain limitations in both time and accuracy. The training process of deep neural networks remains computationally expensive, especially when the model must be frequently retrained to adapt to dynamic network conditions. In addition, the overall accuracy strongly depends on the quality and balance of labeled data. Since the "malicious" class typically represents a small fraction of real traffic, the model may still struggle to maintain high detection accuracy for this class.

Future research should focus on optimizing training time and improving detection accuracy for minority classes through advanced class-weighting, semi-supervised, or transfer learning techniques.

VII. CONCLUSION

In this article, we developed a comprehensive approach to enhancing network monitoring systems through the integration of predictive techniques and machine learning models. The architecture of monitoring systems was thoroughly examined, emphasizing their modular design, which includes monitoring agents, central servers, and alert systems. We compared active and passive monitoring methods, highlighting their advantages and limitations. Additionally, we introduced a novel predictive monitoring framework that anticipates potential network issues before they occur, offering a proactive solution in contrast to traditional reactive methods. Furthermore, the study tackled the issue of data imbalance in machine learning models, which is a significant challenge in accurately detecting anomalies, particularly in underrepresented traffic classes such as malicious activities. To address this, we proposed dynamic class weighting to enhance the accuracy of anomaly detection. The paper also utilized the LUFlow dataset for model training, employing honeypots to gather valuable data for network intrusion detection and demonstrating the practical application of the proposed monitoring framework. The results demonstrate that the proposed transition from binary to multiclass classification provides a more accurate representation of real network traffic conditions. While the numerical metrics appear lower (Precision 0.88, Recall 0.88 compared to 0.999 in binary form), this configuration significantly improves the model's ability to identify the minority malicious class and to differentiate it from uncertain traffic. Thus, the approach enhances the interpretability and operational relevance of intrusion detection models for complex telecommunication environments. In the future, the proposed technique with dynamic adjustment of class weights may become an important step in improving communication network monitoring systems. This method allows for efficient adaptation of the model to new input data, reducing errors in anomaly recognition, which is especially important in complex and changing conditions of real networks. The simplicity of the linear model, which does not require a large number of layers, will not only speed up the process of retraining the system, but also ensure its stability in the event of new types of traffic or threats. This approach opens up prospects for more flexible and reliable predictive monitoring, allowing for increased efficiency and security of modern communications systems.

References

[1] C. L. Aldea, R. Bocu, and R. N. Solca, "Real-time monitoring and management of hardware and software resources in heterogeneous computer networks through an integrated system architecture," *Symmetry*, vol. 15, p. 1134, 2023, https://doi.org/10.3390/sym15061134.

[2] W. Song, M. Beshley, K. Przystupa, H. Beshley, O. Kochan, A. Pryslupskyi, D. Pieniak, and J. Su, "A software deep packet inspection system for network traffic analysis and anomaly detection," *Sensors*, vol. 20, p. 1637, 2020, https://doi.org/10.3390/s20061637.

[3] J. Tang, T. Qin, D. Kong, Z. Zhou, X. Li, Y. Wu, and J. Gu, "Anomaly detection in social-aware IoT networks," *IEEE Trans. Netw. Serv. Manag.*, early access, 2023, https://doi.org/10.1109/TNSM.2023.3242320.

[4] H. Bilakanti, S. Pasam, V. Palakollu, and S. Utukuru, "Anomaly detection in IoT environment using machine learning," *Security Privacy*, 2024, https://doi.org/10.1002/spy2.366.

[5] K. Albulayhi and Q. A. Al-Haija, "Adversarial deep learning in anomaly based intrusion detection systems for IoT environments," *Int. J. Wireless Microw. Technol. (IJWMT)*, vol. 13, no. 4, pp. 1–10, 2023, https://doi.org/10.5815/ijwmt.2023.04.01.

[6] N. Lutsiv, T. Maksymyuk, M. Beshley, O. Lavriv, V. Andrushchak, A. Sachenko, L. Vokorokos, and J. Gazda, "Deep semisupervised learning-based network anomaly detection in heterogeneous information systems," *CMC-Comput. Mater. Continua*, vol. 70, pp. 413–431, 2022, https://doi.org/10.32604/cmc.2022.018773.

[7] O. Aslanli, "Cloud and on-premises based security solution for industrial IoT," *Int. J. Inf. Eng. Electron. Bus. (IJIEEB)*, vol. 16, no. 5, pp. 55–62, 2024, https://doi.org/10.5815/ijieeb.2024.05.02.

[8] S. Lehominova, Y. Shchavinsky, T. Muzhanova, D. Rabchun, and M. Zaporozhchenko, "Application of sentiment analysis to prevent cyberattacks on objects of critical information infrastructure," *Int. J. Comput.*, vol. 22, no. 4, pp. 534–540, 2023, https://doi.org/10.47839/ijc.22.4.3362.

[9] M. Beshley, N. Kryvinska, and H. Beshley, "Quality of service management method in a heterogeneous wireless network using big data technology and mobile QoE application," *Simul. Model. Pract. Theory*, vol. 127, p. 102771, 2023, https://doi.org/10.1016/j.simpat.2023.102771.

[10] G. Nguyen, S. Dlugolinsky, V. Tran, and Á. López García, "Deep learning for proactive network monitoring and security protection," *IEEE Access*, vol. 8, pp. 19696–19716, 2020, https://doi.org/10.1109/ACCESS.2020.2968718.

[11] Y. Chen, H. Peng, L. Huang, J. Zhang, and W. Jiang, "A novel MAE-based self-supervised anomaly detection and localization method," *IEEE Access*, vol. 11, pp. 127526–127538, 2023, https://doi.org/10.1109/ACCESS.2023.3332475.

[12] A. Abdelkhalek and M. Mashaly, "Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning," *J. Supercomput.*, vol. 79, no. 10, pp. 10611–10644, 2023, https://doi.org/10.1007/s11227-023-05073-x.

[13] D. Mahesh and T. S. Kumar, "Machine learning algorithms for detecting DDoS attacks in intrusion detection systems," *Int. J. Wireless Microw. Technol. (IJWMT)*, vol. 14, no. 5, pp. 59–71, 2024, https://doi.org/10.5815/ijwmt.2024.05.05.

[14] X. Li, G. Shi, and Y. Wu, "Utilizing machine learning techniques for network traffic anomaly detection," *Appl. Comput. Eng.*, vol. 36, no. 1, pp. 242–247, 2024, https://doi.org/10.54254/2755-2721/36/20230454.

[15] S. Dong, H. Su, and Y. Liu, "A-CAVE: Network abnormal traffic detection algorithm based on variational autoencoder," *ICT Express*, vol. 9, no. 5, pp. 896–902, 2023, https://doi.org/10.1016/j.icte.2022.11.006.

[16] Z. Hu, R. Odarchenko, S. Gnatyuk, M. Zaliskyi, A. Chaplits, S. Bondar, and V. Borovik, "Statistical techniques for detecting cyberattacks on computer networks based on an analysis of abnormal traffic behavior," *Int. J. Comput. Netw. Inf. Secur. (IJCNIS)*, vol. 12, no. 6, pp. 1–13, 2020, https://doi.org/10.5815/ijcnis.2020.06.01.

[17] I. Zavushchak, "The impact of artificial intelligence on cybersecurity and data protection," *Int. J. Wireless Microw. Technol. (IJWMT)*, vol. 15, no. 4, pp. 65–72, 2025, https://doi.org/10.5815/ijwmt.2025.04.05.

[18] Y.S. Ndichu, S. McOyowo, H. Okoyo, and C. Wekesa, "Detecting remote access network attacks using supervised machine learning methods," *Int. J. Comput. Netw. Inf. Secur. (IJCNIS)*, vol. 15, no. 2, pp. 48–61, 2023, https://doi.org/10.5815/ijcnis.2023.02.04.

[19] M. Goyal and Q. H. Mahmoud, "A systematic review of synthetic data generation techniques using generative AI," *Electronics*, vol. 13, p. 3509, 2024, https://doi.org/10.3390/electronics13173509.

[20] A. Khandare and A. S. Alvi, "Performance analysis of improved clustering algorithm on real and synthetic data," *Int. J. Comput. Netw. Inf. Secur. (IJCNIS)*, vol. 9, no. 10, pp. 57–65, 2017, https://doi.org/10.5815/ijcnis.2017.10.07.

[21] V. Kumar and D. Sinha, "Synthetic attack data generation model applying generative adversarial network for intrusion detection," *Comput. Secur.*, vol. 125, p. 103054, 2023, https://doi.org/10.1016/j.cose.2022.103054.

[22] S. Sanshi, R. Vatambeti, R. V., and S. Z. Rahman, "An efficient optimized neural network system for intrusion detection in wireless sensor networks," *Int. J. Comput. Netw. Inf. Secur. (IJCNIS)*, vol. 16, no. 6, pp. 83–94, 2024, https://doi.org/10.5815/ijcnis.2024.06.07.

[23] B. Rusyn, O. Lutsyk, R. Kosarevych, T. Maksymyuk, and J. Gazda, "Features extraction from multi-spectral remote sensing images based on multi-threshold binarization," *Sci. Rep.*, vol. 13, no. 1, p. 19655, 2023, https://doi.org/10.1038/s41598-023-46785-7.

[24] X. Ma and W. Shi, "AESMOTE: Adversarial reinforcement learning with SMOTE for anomaly detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 943–956, 2021, https://doi.org/10.1109/TNSE.2020.3004312.

[25] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32464–32476, 2020, https://doi.org/10.1109/ACCESS.2020.2973730.

[26] A. A. Awad, A. F. Ali, and T. Gaber, "An improved long short term memory network for intrusion detection," *PLoS One*, vol. 18, no. 8, p. e0284795, 2023, https://doi.org/10.1371/journal.pone.0284795.

[27] K. Mounika, P. V. Rao, and A. Anbalagan, "Modified CNN model for network intrusion detection and classification system using local outlier factor-based recursive feature elimination," *Int. J. Comput. Netw. Inf. Secur. (IJCNIS)*, vol. 17, no. 1, pp. 82–91, 2025, https://doi.org/10.5815/ijcnis.2025.01.07.

[28] A. I. Jony and A. K. B. Arnob, "Securing the Internet of Things: Evaluating machine learning algorithms for detecting IoT cyberattacks using CIC-IoT2023 dataset," *Int. J. Inf. Technol. Comput. Sci. (IJITCS)*, vol. 16, no. 4, pp. 56–65, 2024, https://doi.org/10.5815/ijitcs.2024.04.04.

[29] A. Sharma and H. Babbar, "LUFlow: Attack detection in the Internet of Things using machine learning approaches," *Proceedings of the 2023 International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, Ballar, India, 2023, pp. 1-5, https://doi.org/10.1109/ICDCECE57866.2023.10150813.

[30] V. Kosheliuk and Y. Tulashvili, "Implementing honeypots for detecting cyber threats with AWS using the ELK," *Int. J. Comput.*, vol. 23, no. 4, pp. 618–624, 2024, https://doi.org/10.47839/ijc.23.4.3761.

[31] Google, "Datasets: Imbalanced datasets," Google for Developers. [Online]. Available at: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data.

[32] Keras, "Imbalanced classification: credit card fraud detection," Keras.io. [Online]. Available at: https://keras.io/examples/structured_data/imbalanced_classification/.

[33] S. A. Wahab, S. Sultana, N. Tariq, M. Mujahid, J. A. Khan, and A. Mylonas, "A multi-class intrusion detection system for DDoS attacks in IoT networks using deep learning and transformers," *Sensors*, vol. 25, no. 15, p. 4845, 2025, https://doi.org/10.3390/s25154845.

[34] H. Kamal and M. Mashaly, "Robust intrusion detection system using an improved hybrid deep learning model for binary and multi-class classification in IoT networks," *Technologies*, vol. 13, no. 3, p. 102, 2025, https://doi.org/10.3390/technologies13030102.

[35] A. K. Sharma, R. Gupta, and P. Singh, "Multiclass classification by various machine learning techniques," *Math. Probl. Eng.*, vol. 2023, pp. 1–11, 2023, https://doi.org/10.1155/2023/1956865.

[36] F. Ahmad Khan, A. Ali Shah, N. Alshammry, S. Saif, Wasim Khan, M. O. Malik, Z. Ullah, "Balanced multi-class network intrusion detection using machine learning," *IEEE Access*, vol. 12, pp. 178222-178236, 2024, https://doi.org/10.1109/ACCESS.2024.3503497.

[37] S.-M. Tseng, Y.-Q. Wang, and Y.-C. Wang, "Multi-class intrusion detection based on transformer for IoT networks using CIC-IoT-2023 dataset," *Future Internet*, vol. 16, no. 8, p. 284, 2024, https://doi.org/10.3390/fi16080284.

ANDRIY LUTSIUK Postgraduate student of the Department of Electronics and Information Technology at Lviv Polytechnic National University. His research interests: artificial intelligence, machine learning, network monitoring.

OREST LAVRIV Doctor of science, Lecturer of the Department of Electronics and Information Technology at Lviv Polytechnic National University. His research interests: systems architecture, 5G, SDN, IoT routing cloudification, cloud.

MYKOLA BESHLEY is currently a Professor in the Information and Communication Technologies Department at Lviv Polytechnic National University. His current research interests include next-generation IoT, cloud computing, big data, software-defined networks, intent-based networks, network security, artificial intelligence, heterogeneous networks, and 5G and 6G technologies.

MYKOLA BRYCH is a Ph.D. in Technical Sciences, Associate Professor of the Information and Communication Technologies Department at Lviv Polytechnic National University. Research interests: 5G mobile networks, data transmission in heterogeneous networks, UAV, networking.