The 14th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2024
11-13 October, 2024, Athens, Greece

Development of Network Traffic Monitoring System Elements Using Deep Learning

Vasyl Melnyk
Volyn Professional College of the National University of Food Technologies, Lutsk, Ukraine
melnykvasyl@yahoo.com

Nataliia Bahniuk
Department of Computer Engineering and Security, Lutsk National Technical University, Lutsk, Ukraine
bahniuk_nataliia@lutsk-ntu.com.ua

Kateryna Bortnyk
Department of Computer Engineering and Security, Lutsk National Technical University, Lutsk, Ukraine
katerina.bortnyk@gmail.com

Inna Kondius
Dean of Faculty of Computer and Information Technologie, Lutsk National Technical University, Ukraine
Innastk@ukr.net

Nataliia Zubovetska
Department of Applied Mechanics and Mechatronic, Lutsk National Technical University, Lutsk, Ukraine
n.zubovetska@lntu.edu.ua

Kostiantyn Kondius
Department of Software Engineering Lutsk National Technical University. Lutsk, Ukraine
kondius_kostya@ukr.net

Abstract— in this article we develop a network traffic monitoring system elements using deep learning that demonstrates the high effectiveness of deep learning in detecting and analyzing network traffic, which contributes to ensuring the security and stability of corporate networks. In this context we develop algorithms for detecting and countering potential threats in network traffic, providing deeper analysis and effective response to cyber threats.

Keywords— Deep Learning, Monitoring, Software Tool, Network Traffic

I. INTRODUCTION

The development of information technologies has led to a growing interest in researching methods of protecting corporate networks in the context of today's digital world, where corporate networks are becoming increasingly complex and distributed, and data protection is a key priority.

Nowadays, many studies have focused on the use of cloud technologies and deep learning to secure network traffic, but there is a need for further development and optimization of these systems. The importance of this research topic lies in the need to develop effective means of protection and monitoring of network traffic in corporate networks. Network traffic monitoring is an important component of modern information technologies and infrastructures. With the development of communication technologies and the large growth of network traffic volumes, there was a need for more advanced and intelligent more powerful and versatile monitoring tools, are used to provide detailed information about network traffic, as well as to detect network security threats and remediate its work in real time.

Network traffic monitoring, defined as the process of monitoring and analyzing network traffic to ensure network reliability, performance, and security, encompasses a wide range of technologies and concepts. Due to historical development and the introduction of new technologies, network traffic monitoring has become a critical aspect of maintaining networks and ensuring their security and performance in today's information environment.

Therefore the main goal of the research is to design and analyze a secure corporate network using cloud infrastructure and Deep Learning methods to monitor and protect network traffic. Research tasks include developing monitoring models, threat detection algorithms, researching the effectiveness of cloud technologies, designing a secure network architeecture, and analyse strategies to improve security. At the same time the development of cloud technologies and Deep Learning (DL) opens up new opportunities for creating reliable security systems. In this context DL is powerful effective tool for different applications but it is raised the concern about privacy and security questions, for example [1] presents a good overview about techniques in DL privacy, security, and defense. Therefore DL models must not provide sensitive and private user information [2-3], although training process enables DL models to analyze high-level abstractions and extract important information from raw data, if image and speech recognition , natural language processing is needed [4-6]. Deep Learning is a subfield of machine learning that uses neural networks with layers to solve complex problems [7] . In monitoring network traffic, Deep Learning technologies have found wide application, contributing to the improvement of analysis and detection of anomalies.

In [8] robust Machine Learning Systems are studied, Reliability and Security for Deep Neural Networks machine learning is used for advanced data, here analytics and intelligent control are based upon machine learning algorithms. Reliability aspect is also studied in [9-11], for example prominent types of reliability threats, i.e. soft errors, aging, and process variations, it refers to the resilience against vulnerabilities to faults in the hardware/devices.

This paper is organized as following. The problem formulation and methodological base is presented in section II and III. In section IV we present development of network traffic monitoring system, and in section V we describe detection and protection processes against attacks. Recommendation for a future work and conclusion and are finally presented in the last section.

II. STATE OF THE ART

Thus, subject of the research of this paper is a network traffic monitoring system based on deep learning in corporate networks. In this paper Deep Learning is used to detect anomalies in network traffic, such as malicious attacks, viruses, or hardware failures. Deep Learning systems

979-8-3315-1093-0/24/$31.00 ©2024 IEEE

learn from historical data and identify changes in traffic that indicate anomalies. An example would be the detection of DDoS attacks or internal threats in a corporate network. Example: Google uses Deep Learning to detect anomalies in its own network. One of their tools, B4, uses Deep Learning to detect anomalous network traffic routes [12]. Deep Learning predicts network load based on historical data and internal and external factors. Scientific novelty and actuality of the research lies in the development of complex solutions for the protection of corporate networks using advanced technologies that meet modern security and efficiency requirements.

In this work we make the comparison with recent research included [13-18] that develop different strategies for optimizing and improving network security have been proposed and implemented, which allows for effective resistance to various cyber threats. For example, in [14] deep learning approach to mitigate DDoS attacks in SDN was done. In [15] new framework for DDoS attack detection and defense in SDN environment was developed, and in [16] authors propose an intrusion detection systembased on combining cluster centers and nearest neighbors, in [17] authors develop the DBN-LSTM attack method in order to detect and prevent DDoS attacks in networks, including Generative Adversarial Networks, Deep Belief Networks and Long Short-Term Memory using semi-supervised deep learning models, leverage unlabeled data for pretraining deep architectures. In [18] it was implemented a methodology of DDoS attacks treatment by developing a hybrid AR-MLP method provides the appropriate good results by verifyng the most important options with the human aid.

These results may be of interest to enterprises for their employees who are engaged in activities in the field of computer networks and cloud technologies and wish to improve the state of security of their networks by using solutions based on artificial intelligence, because we offer a simple and scalable solution for analyzing network traffic, which is technically easy to integrate into any new or existing project, and examine the impact of training data for a neural network and the accuracy of its decisions and, as a result, the security of the system.

III. Objective of the Article

The objective of the paper is the development and implementation of a network traffic monitoring system in the cloud infrastructure using Deep Learning technologies. In this paper we create an effective tool for analyzing and optimizing corporate networks operating in a cloud environment. So, we design a network traffic monitoring model that demonstrates the high effectiveness of deep learning in detecting and analyzing network traffic, which contributes to ensuring the security and stability of corporate networks, we develops algorithms for detecting and countering potential threats in network traffic, providing deeper analysis and effective response to cyber threats.

We contribute to increasing the level of productivity of corporate networks using cloud infrastructure. Improving network security and the ability to detect anomalies and potential threats, reducing the risk of network incidents and increasing the reliability of its operation, providing network operators and administrators with tools to analyze and optimize network traffic.

To do this we develop the monitoring system, which includes components for collecting and analyzing network traffic, implementation of Deep Learning technologies to detect anomalies in network traffic and automatically react to them, providing the ability to visualize and analyze monitoring results for network operators and administrators. We develop of security mechanisms to protect data and infrastructure of the monitoring system, make testing and debugging the system in real conditions.

IV. Basic Concepts And Research Methodology

A. Existing Solutions and Analysis

In this section the advantages and disadvantages of using cloud infrastructure in corporate networks were studied, the key factors affecting the performance and security of networks were identified.

Today, there are many solutions for monitoring network traffic using Deep Learning technologies. Here are some examples of existing solutions and their analysis:

1. *Kentik Detect* is a network traffic monitoring platform that uses Deep Learning to detect anomalies and optimize the network. The system analyzes large volumes of data and uses neural networks to automatically detect anomalies in real time. Kentik Detect enables network operators to respond to anomalies quickly and efficiently. Advantages: ability to analyze large volumes of data in real time; automatic detection of anomalies and development of analytical reports; optimizing network bandwidth based on data analysis.

2. *Darktrace* is a cyber security platform that uses Deep Learning technologies to detect cyber threats and anomalies in network traffic. Darktrace uses neural networks to analyze the behavior of users and devices on the network and detect anomalous activities. Advantages are: ability to detect 0-day threats and insider attacks; self-learning and ability to adapt to new threats; monitoring of user and device activity in real time.

3. *Cisco ThousandEyes* is a cloud network monitoring solution that uses Deep Learning to analyze network traffic data and presents this information in the form of graphs and reports. It provides detailed insight into the operation of the cloud infrastructure. Each of these solutions demonstrates the power of Deep Learning in network traffic monitoring. They provide innovative capabilities for anomaly detection, network optimization, and cloud security. These solutions help enterprises maintain the highest level of performance and reliability of their corporate network.

Thus, to implement the given task, we use Deep Learning, as it opens wide opportunities for optimizing corporate networks in the cloud environment, offering a number of innovative solutions to improve network performance and security. One of the most important aspects of the potential of Deep Learning is its ability to analyze and use large volumes of data, which is becoming increasingly important in cloud environments. Deep Learning can be used to optimize routing by analyzing large amounts of traffic data, thereby determining the best routes for different types of traffic and different locations. This can significantly improve performance and reduce latency, which are critical

aspects for corporations with large geographic footprints and high network demands.

B. Methodology Based on Python

The developed methodology of monitoring of Network Traffic is based on Python, because its is a powerful programming language widely used in the development of network monitoring software. Its ease of reading and writing, as well as the large number of libraries, make it an ideal choice for developing complex monitoring systems. Python can be used to collect, process, and analyze network data, as well as to integrate with other tools and APIs.

Python is considered one of the most popular programming languages for machine learning and deep learning, due to the following advantages:

– Machine learning libraries: Python has a rich set of machine learning libraries, such as TensorFlow, Keras, PyTorch, Scikit-learn, that facilitate the development and implementation of deep learning models.

– Flexibility and Intuitiveness: Python is known for its readability and simplicity of syntax, which makes it accessible even to non-expert programmers.

– Community and Support: A large developer community and many training and support resources make Python the #1 choice for many machine learning professionals.

– Integration with other tools: Python easily integrates with various systems and APIs, which allows it to be effectively used in complex network monitoring systems.

D. Data Preparation and Processing

To effectively use Deep Learning in monitoring network traffic, it is necessary to properly prepare and process log data, because of this the developed method depending to goal is able to use following databases: MongoDB database that can be useful for storing and processing unstructured network traffic data; MySQL as a relational database with a wide range of possibilities for processing structured data; PostgreSQL database with high reliability and extensibility for working with complex queries; InfluxDB that specializes in storing information that changes over time and can be useful for data monitoring.

After data collecting the data preparation and processing take place. Nginx logs contain valuable information about network activity, but in order to analyze them using machine learning algorithms, they must first be converted into a structured format such as JSON. This process involves parsing text logs and converting them into a format suitable for further processing by a neural network with aid of data parsing and processing methods. Nginx logs typically contain data in a format that includes the client's IP address, timestamp, HTTP method, request URL, response status code, response size, and other parameters. To convert this data into a format suitable for processing, we use Python scripts that parse the text of the logs and convert them into JSON format.

.

V. DEVELOPMENT OF NETWORK TRAFFIC MONITORING SYSTEM

A. Choosing Necessary Tools

In addition to choosing a programming language, framework, database and infrastructure, it is also important to choose other necessary tools for developing a network traffic monitoring system using Deep Learning technologies. Below there are additional tools that can be used within the project.

We use the TensorFlow and Keras libraries, and the pytest and unittest libraries to automate Python code testing. We also use the PyCryptodome library to implement AES (Advanced Encryption Standard) or DES (Data Encryption Standard). These encryption methods provide strong security in situations where fast processing of large volumes of data is required, or for secure data transmission over a network, used in financial transactions and other situations where speed and security are important.

Choosing an integrated development environment is an important step. In this project we use PyCharm for Python development, and IntelliJ IDEA for Java. Integrated environments provide a convenient interface for programmers and tools for debugging code.

Using logging tools helps us to monitor and analyze system performance. Debugging tools such as `pdb` for Python allow us to find and fix errors in your code. Using system performance monitoring tools helps to identify and optimize speed and resource issues; using Jenkins helps us to automate the design, test, and deployment processes of a system. Monitoring tools such as Prometheus is used to monitor the health of servers and infrastructure.

In order to monitoring and analysis of user behavior we use Deep Learning technologies which help to detect unusual user activity on the network, which may be a sign of an insider threat. Protection against application security threats has been implemented by audit and penetration testing and application security in development. Regular auditing and penetration testing of monitoring system applications allow us to identify and fix potential vulnerabilities. We apply the security best practices when developing monitoring system applications to prevent application-level attacks. Defending against network security attacks in a corporate cloud infrastructure is a complex task that requires a comprehensive approach. The use of firewalls, IDPS, access control systems, network infrastructures with redundancy and cloud services to protect against DDoS attacks, as well as monitoring and analyzing user behavior, helps to ensure a high level of security in network monitoring.

VI. DESIGN OF BASIC SYSTEM PART USING DEEP LEARNING

Designing the basic part of the network traffic monitoring system using Deep Learning is a crucial aspect of ensuring reliable and productive functioning of the system in the context of corporate cloud infrastructure. Deep Learning allows you to automatically detect abnormal activity and attacks in the network, as well as optimize its operation.

A. System Architecture

The main part of the system is the Deep Learning module, which performs network traffic analysis. The use of deep neural networks allows for the detection of patterns and anomalies that may indicate problems in the network.

To achieve the best efficiency in the network traffic monitoring system, it is important to choose the appropriate Deep Learning algorithms.

The main algorithm include development of the following networks:

Convolutional Neural Networks (CNN): Used to analyze structured data such as network traffic. CNNs are distinguished by their ability to detect important features in data without the need to manually identify them. In network traffic, CNNs can detect complex patterns and dependencies that can help predict network load, detect anomalies, or ensure network security. This is particularly useful in the context of big data and IoT, where large volumes of data from various devices and sources need to be analyzed.

Recurrent Neural Networks (RNN): Used to analyze sequential data such as network packets. This is because RNNs are designed to maintain an internal memory that allows them to take into account information from previous steps in the sequence. In the context of network traffic, this means that RNNs can detect dependencies and patterns in sequences of network packets, which is key for tasks such as network load prediction, anomaly detection, or security analysis. RNNs are particularly useful for processing data where context and sequence are important

B. Collection and Data Pre-Processing

feeding the data to the input of the Deep Learning module, they are pre-processed to reduce the volume and improve the quality of the data. It is important to carefully plan the process of data collection and preprocessing, as it affects the accuracy and performance of the system (Fig. 1).

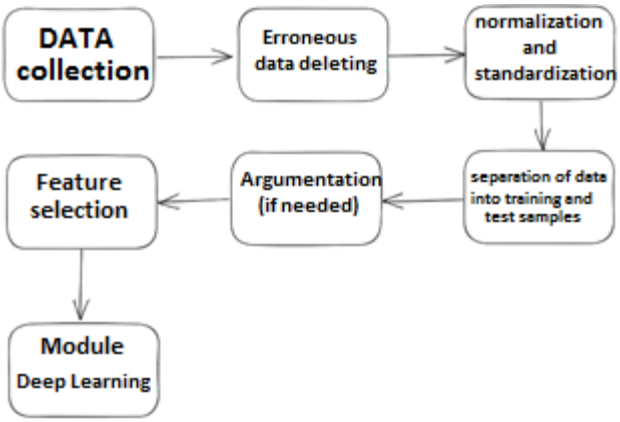

Fig. 1. Developed data processing scheme

Tuning and optimization of Deep Learning models are realized as follows. To achieve optimal results, it is important to tune and optimize Deep Learning models. This may include the selection of hyperparameters, the use of model ensembles, and the development of performance tracking metrics.

One of the practical applications example of Deep Learning in the monitoring system is the detection of DDoS attacks. A Deep Learning model can learn to recognize typical patterns of network behavior and deny anomalous activity that could be a sign of a DDoS. This allows automatic detection and response to attacks, ensuring network security.

VII. SOFT CONFIGURATION AND EXPERIMENTAL TESTS

Installing and configuring Nginx on EC2 instances in AWS can be automated using Terraform. Terraform allows you not only to create the necessary infrastructure components, but also to automate the process of installing and configuring software on these components. For this, the concept of "user data" in EC2 is used, which allows you to execute scripts when the instance is launched

A. Nginx configuration

After installing Nginx, you need to configure it to handle incoming traffic and collect logs. The Nginx configuration file, usually located in /etc/nginx/nginx.conf, can be configured to provide the desired functionality. An example configuration can be seen in Fig.2.

```
# nginx.conf.tpl
http {
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    error_log /var/log/nginx/error.log;

    #
}
```

Fig.2. Example nginx configuration

File templates can be used to pass Nginx configuration through Terraform. Terraform allows you to generate configuration files based on templates and push them to EC2 instances as they are created. To do this, you can use the `template_file` resource and the user_data variable in Terraform. To generate a configuration file, it is necessary to define start by defining the template_file resource.

To transfer a configuration file to an EC2 instance, it is necessary to transfer it through the user_data directive.

B. Data preparation and processing

To effectively use deep learning in monitoring network traffic, it is necessary to properly prepare and process log data. Consider appropriate example.

Example (parsing Nginx logs).

The Python script is shown on Fig.3 demonstrates how to read Nginx logs, parse them, and convert them into structured JSON format.

```
import re
import json

#
log_pattern = re.compile(
    r'(?P<ip>\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}) - - '
    r'\[(?P<datetime>[^\]]+)\] '
    r'"(?P<method>\S+) (?P<path>\S+) \S+" '
    r'(?P<status>\d{3}) (?P<size>\d+) '
    r'"(?P<referer>[^"]+)" "(?P<user_agent>[^"]+)"'
)

def parse_log_line(line):
    match = log_pattern.match(line)
    if match:
        return match.groupdict()
    return None

#
log_data = []
with open('access.log', 'r') as file:
    for line in file:
        parsed_line = parse_log_line(line)
        if parsed_line:
            log_data.append(parsed_line)

#
json_data = json.dumps(log_data, indent=4)
print(json_data)
```

Fig.3. An example of a script for parsing Nginx logs

This example uses a regular expression to parse each line of the log. The `parse_log_line` function parses each line and returns a dictionary of data. After parsing all the strings, the data is converted to JSON using `json.dumps`, making it suitable for neural network processing.

Before training the model, we need to prepare `Nginx log` data that has been converted into JSON format. This includes vectorization of text data and normalization of numeric values. Nginx logs contain valuable information about network activity, but in order to analyze them using machine learning algorithms, they must first be converted into a structured format such as JSON. This process involves parsing text logs and converting them into a format suitable for further processing by a neural network (Fig.4).

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam

#
model = Sequential([
    Dense(128, activation='relu', input_shape=(input_shape,)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

#
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Fig.4. Neural network model

After the model is built, it is trained on the training data and then tested on the test set (Fig.5).

```
#
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

#
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f" Accuracy: {test_accuracy}")
```

Fig.5. Neural network model

Data parsing and processing methods. Nginx logs typically contain data in a format that includes the client's IP address, timestamp, HTTP method, request URL, response status code, response size, and other parameters. To convert this data into a format suitable for processing, you can use Python scripts that parse the text of the logs and convert them into JSON format.We use a regular expression to parse each line of the log.

The `parse_log_line` function parses each line and returns a dictionary of data. After parsing all the strings, the data is converted to JSON using json.dumps, making it suitable for neural network processing. Before training the model, you need to prepare Nginx log data that has been converted to JSON format. This includes vectorization of text data and normalization of numeric values.

C. Model development, training and evaluation

After processing the log data, we create a neural network model using TensorFlow and Keras (Fig.4,5).

The developed model can be integrated into the cloud infrastructure. The effectiveness of using the developed deep learning model for network traffic monitoring depends greatly on its integration with cloud infrastructure. This process involves moving the model and log parsing script to the cloud infrastructure, setting up automated traffic monitoring, and implementing alerts when potential threats are detected.

Model integration includes the following stages:

1) *Deployment of EC2 Instances through Terraform*. To the current configuration, it is necessary to add the installation of the Python environment and the transfer of the learning model itself to the server. To transfer scripts from a local disk to an EC2 instance as part of the Terraform configuration, you can use the Terraform provider for AWS and the aws_instance resource. This can be done using the file provisioner in Terraform, which allows you to copy files from a local machine to a remote instance.

2) *Setting up* a cron task to regularly execute a log parsing script and check them with a machine learning model.

3) *Sending notifications.* If potentially dangerous traffic is detected, the script should respond

4) *Sending notifications*. If potentially dangerous traffic is detected, the script should send a notification to e-mail or messenger.

With aid of widely used (see for example [14]) evaluation metrics we estimate the effectiveness of developed methods of intrusion detection. The metrics are obtained from various attributes, that provide the information about actual and predicted data and includes the following:

- True Positive (TP): Instances of data correctly classified as an attack
- False Negative (FN): Data instances incorrectly predicted as normal instances
- False Positive (FP): Instances of data incorrectly classified as an attack
- True Negative (TN): Instances correctly classified as normal instances.

For real-time traffic analysis, the appropriate confusion matrix [14,19] indicates the correct and wrong predictions respectively of a particular classifier. In software environment, this matrix can make it possible to estimate the reliability of the attacs detection system, make decisions about implementing security measures taking into account the observed errors [19]. In this context the confusion matrix is used in order to attack prediction, to identify the errors types, to make the decision if the model needs certain tuning, additional acts or data. Based on above, we define the precision as a ratio of correctly identified attacks to the general number of instances identified as attacks:

$$\propto (precision) = \frac{TP}{TP + FP}$$

Analysis of model accuracy depending on the amount of training data has been executed in following manner. Data samples with different number of logs were used to compare the accuracy of the machine learning model.

We use different training data for the models and their volumes are following: 40; 70; 100; 1000; 10000; 100000; 10^6. Thanks to this, it was possible to determine the values that correspond to the necessary amount of information, which is sufficient to obtain "answers" of satisfactory accuracy from the model. The results of model training and relative accuracy on data samples of different volume are given on Fig.6.

From this diagram, it is clear that for satisfactory model accuracy (90% or more), we provide for training at least 1000 log data objects as training data. If possible, it is better to use samples of a much larger volume, namely tens or hundreds of thousands of objects. This will ensure the best accuracy of the model.

In this work we make the comparison with recent research included [14-18] which took the NSL-KDD and

CIC-IDS 2019 and Mendeley 2020 datasets, that are the data for investigations into intrusion identified by the Canadian cyber security institute. These datasets include qualitative and quantitative data on traditional attacks, they are used for network traffic analysis, including protocols and attacks over time, targets and the target IPs, and location port. Authors [14] focused on DDoS traffic and use it to DDoS attack detection, they used following material and system specifications: system manufacturer Lenovo, processor Intel core i7 6700 CPU with 3.4 GHz, memory 8GB, operating system Ubuntu 16.04, emulator Mininet 2.2.1; a total of 90,000 samples were uploaded for distribution, with a 60:40 split between malicious traffic and DDoS. The studied data was split for training (80%) and testing purposes (20%) with aid of train test classification methods based on the library scikit, the test size was = 0.02. The data sample of 71000 were used for training and 19000 samples for testing.

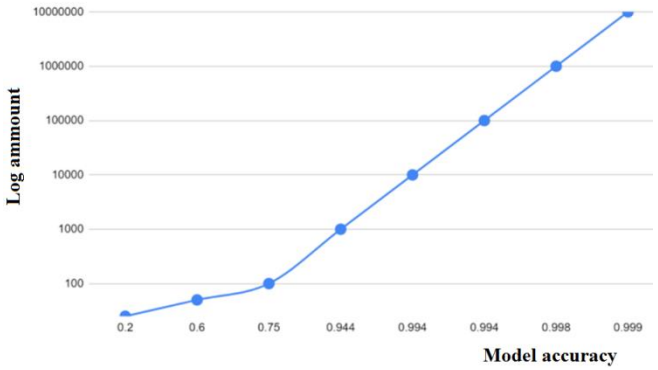

Fig.6. Dependence of model accuracy on the amount of training data

According to the results presented in Table I the avarage value of detection time is equal 0.04, as shown on Fig.8, the average train time is equal 35,13, as illustrated in Fig.9.

TABLE I. COMPARISON OF DEVELOPED RESULT WITH CURRENT RESEARCH CONTRIBUTIONS

Existing results, number of contribution (references)	Precision	Detection time, min	Avarage time	Train time, min	Avarage time
1 [14]	0,9841	0.029	0,04	23,00	35,13
2 [15]	0,9810	0,061		39,52	
3 [16]	0,9820	-		40,00	
4 [17]	0,9644	-		-	
5 [18]	0,9791	-		-	
6, Our result	0,9900	0,030		38,00	

On the Table I we give a detail comparison of different recent approaches to attacks detection. From Fig.7 it can be seen that our approach is giving successful efficiency in classifying the data packets as attack and precision. The time required for attack detection is also minimal (see Fig. 8) as compared to other existing approaches and CPU utilization is minimal hence leading to saving of resources and generating better results. Figure 9 shows the comparison of our approach with the existing works with respect to train times, which is also appropriate in our method.

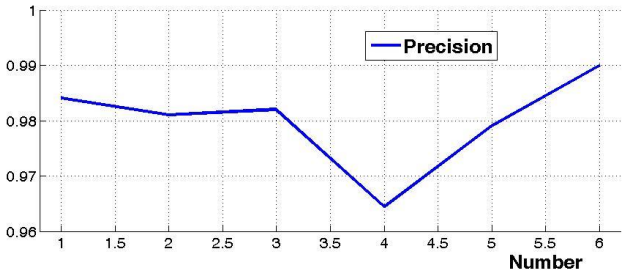

Fig.7. Precision Comparison of developed result with current research contributions

Proposed method is siutable for analyzing of extensive datasets, making them convenient for analyzing large volumes of traffic data, satisfied comprehensive processing and easier data detection.

Fig.8. Comparison of detection times

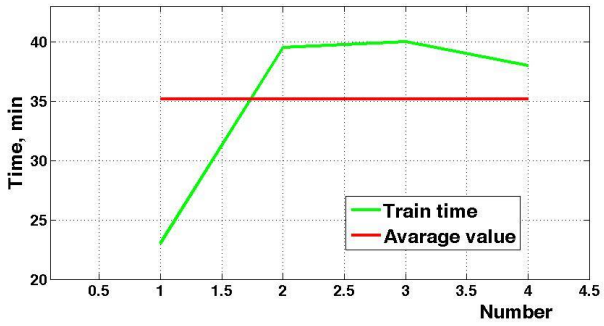

Fig.9. Comparison of train times

Using the available MatLab tools we calculate the values of assessments. In order for predicting and analyze the unknown predicted values we make the mathematical modeling using MatLab appropriate tools. On the Fig.10 the plotting of the studied datasets is presented. For this purpose we make fit in curve of the loaded data using the appropriate Curve Fitting Toolbox, and obtain the polynomial for interpolation and characterize data with aid of a global fit in order to obtain a simple empirical models.

The main advantages of selected polynomial fit is reasonable flexibility for data. Analyzing the data-set (Fig.5) we obtain appropriate resulting linear model which is a polynomial of the degree 2 in a following form:

$$F(x) = p_1*x^2 + p_2*x + p_3$$

Coefficients (with 95% confidence bounds): $p_1 = -8.751e\text{-}10$ (-1.216e-09, -5.341e-10); $p_2 = 2.99e\text{-}05$ (2.192e-05, 3.788e-05); $p_3 = 0.7513$ (0.7221, 0.7806).

The calculating efficiency are following. GOF is equal: SSE: 0.0017; RSQUARE: 0.9729; DFE: 6; ADJRSQUARE: 0.9639; RMSE: 0.0169.

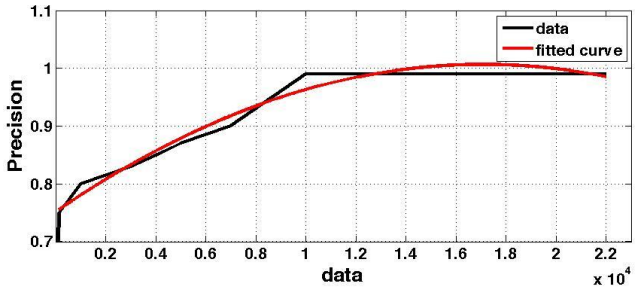

Fig.10. Polynomial modeling of examined dataset

Optimizing and improving infrastructure of the developed system that use deep learning to monitor network traffic are important aspects of improving the efficiency and reliability of a corporate network.

Cloud infrastructure optimization in further research may include the following stages:

1) *Resource scaling.* Provision of sufficient computing resources and memory for EC2 instances to efficiently process large volumes of data and requests.

2) *Scaling automation*. Using AWS Auto Scaling to automatically scale the number of instances based on the current system load.

Expanding security and reliability measures:

– *Implementation of secure communication channels*. Using encryption to protect data in transit between cloud infrastructure and end users.

– *Data backup*. Regular backup of data and models to prevent loss of important information.

– *System monitoring*. Implementation of a monitoring system for tracking the state of cloud infrastructure and a deep learning model.

Implementing these optimization and improvement strategies can significantly improve the efficiency, reliability, and security of a deep learning-based network traffic monitoring system. This, in turn, will contribute to better detection and response to potential threats in corporate networks.

VIII. CONCLUSIONS

We have developed the network traffic monitoring system using Deep Learning technologies in the cloud infrastructure, the developed model demonstrates the high efficiency of deep learning in the detection and analysis of network traffic, which contributes to the security and stability of corporate networks.

In our future research we plan to improve the developed results by fine-tuning hyperparameters, use of extended data, application of other deep learning models. Conducting a careful search for the optimal hyperparameters of the model, including the learning rate, the number of neurons in the hidden layers, and the types of activation functions, can significantly improve the accuracy of the model. Increasing the volume and variety of log data can help models better generalize and detect more complex patterns in network traffic. Experimenting with different deep neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), can reveal more efficient approaches for a particular type of data.

REFERENCES

[1] S.D. Afrah, K.H. Noor. A review of Deep Learning Privacy, Security and Defenses. Technium Vol. 12, pp.65-83 (2023) ISSN: 2668-778X.

[2] M. S. Riazi, B. D. Rouani, and F. Koushanfar, “Deep Learning on Private Data,” IEEE Secur Priv, vol. 17, no. 6, pp. 54–63, Nov. 2019, doi: 10.1109/MSEC.2019.2935666.

[3] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, “A Survey of Deep Learning Methods for Cyber Security,” Information 2019, Vol. 10, Page 122, vol. 10, no. 4, p. 122, Apr. 2019, doi: 10.3390/INFO10040122.

[4] A. S. Dawood, “Machine Learning and Artificial Neural Network for Data Mining Classification and Prediction of Brain Diseases,” International Journal of Reasoning-based Intelligent Systems, vol. 1, no. 1, p. 1, 2023, doi: 10.1504/IJRIS.2023.10052940.

[5] B. Shickel, P. Tighe, others. “Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis,” ieeexplore.ieee.org, Accessed: Mar. 15, 2023. Available: https://ieeexplore.ieee.org/abstract/document/8086133/

[6] M. I. Tariq et al., “A Review of Deep Learning Security and Privacy Defensive Techniques,” Mobile Information Systems, vol. 2020, 2020, doi: 10.1155/2020/6535834.

[7] Y. Bengio, Y. Lecun, G. Hinton. “Deep learning for AI. Communications of the ACM”. 2021. T. 64, № 7. pp. 58–65. URL: https://doi.org/10.1145/3448250 (дата звернення: 18.08.2023).

[8] M. A. Hanif, F. Khalid, R. Vidya, W. Putra, S. Rehman, M. Shafique. "Robust Machine Learning Systems: Reliability and Security for Deep Neural Networks". Conference Paper, 2018 DOI: 10.1109/IOLTS.2018.8474192

[9] R. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies”, IEEE TDMR, vol. 5, no. 3, pp.305-316, 2005.

[10] S. Rehman, “Reliable Software for Unreliable Hardware – A Cross-Layer Approach”, PhD Thesis, (150715/01), KIT, Karlsruhe, 2015.

[11] K. Kang et al., “NBTI induced performance degradation in logic and memory circuits: how effectively can we approach a reliability solution?”, in ASPDAC, 2008.

[12] Y. Xin and others. “Machine Learning and Deep Learning Methods for Cybersecurity” IEEE Access. 2018. T. 6. C. 35365–35381. URL: https://doi.org/10.1109/access.2018.2836950.

[13] J. Schmidhuber, “Deep learning in neural networks: An overview,” Elsevier, 2015.

[14] H. S. Dhadhal et al. “Deep Learning Approach to Mitigate DDoS Attacks in SDN”. Journal of Computer Science, 2024, no. 20 (9): 1030.1039 DOI: 10.3844/jcssp.2024.1030.1039

[15] L.Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, & Y. Deng. “A New Framework for DDoS Attack Detection and Defense in SDN Environment”. 2020, IEEE Access,no. 8, pp. 161908–161919.

[16] W.-C. Lin, S.-W. Ke, & C.-F. Tsai. “CANN: An intrusion detection system based on combining cluster centers and nearest neighbors”. Knowledge-Based Systems, no. 78, pp.13–21. https://doi.org/10.1016/j.knosys.2015.01.009, 2015.

[17] L.Chen, Z.Wang, R. Huo & T. Huang. “An Adversarial DBN-LSTM Method for Detecting and Defending against DDoS Attacks in SDN Environments”. Algorithms, no. 16(4), 197. https://doi.org/10.3390/a16040197 , 2023.

[18] Y. Wei, J. Jang-Jaccard, F.Sabrina, A. Singh, W.Xu, S. Camtepe. “AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification” IEEE Access, 9, 146810–146821. https://doi.org/10.1109/access.2021.3123791, 2021.

[19] F.M. Salem, H. Youssef, I. Ali, & A. Haggag. A variable-trust threshold-based approach for DDOS attack mitigation in software defined networks. PLOS ONE, 2022, no. 17(8), e0273681. https://doi.org/10.1371/journal.pone.0273681.