Computer Science • 26(SI) 2025 https://doi.org/10.7494/csci.2025.26.SI.7079

Aleksander Ogonowski
Michał Żebrowski
Arkadiusz Ćwiek
Tobiasz Jarosiewicz
Konrad Klimaszewski
Adam Padee
Piotr Wasiuk
Michał Wójcik

PRELIMINARY STUDY ON ARTIFICIAL INTELLIGENCE METHODS FOR CYBERSECURITY THREAT DETECTION IN COMPUTER NETWORKS BASED ON RAW DATA PACKETS

Abstract *Most of the intrusion detection methods in computer networks are based on traffic flow characteristics. However, this approach may not fully exploit the potential of deep learning algorithms to directly extract features and patterns from raw packets. Moreover, it impedes real-time monitoring due to the necessity of waiting for the processing pipeline to complete and introduces dependencies on additional software components. In this paper, we investigate deep learning methodologies capable of detecting attacks in real-time directly from raw packet data within network traffic. Our investigation utilizes the CIC IDS-2017 dataset, which includes both benign traffic and prevalent real-world attacks, providing a comprehensive foundation for our research.*

Keywords cybersecurity, artificial intelligence, intrusion detection system, CNN, LSTM, saliency map, deep learning

Citation Computer Science 26(SI) 2025: 45–68

1. Introduction

The rapid proliferation of digital technologies has transformed various aspects of human life, enabling unprecedented convenience and efficiency. Industries such as finance, healthcare, energy, and transportation have especially benefited from these advancements, achieving remarkable improvements in operational efficiency and service delivery. However, this digital revolution has also introduced new vulnerabilities and threats, especially in the realm of cybersecurity.

For instance, the financial sector has seen a dramatic increase in cyber-attacks aimed at stealing sensitive data or disrupting services [26]. The healthcare industry, which now heavily relies on digital records and connected medical devices, faces threats that could expose patient safety and privacy [17]. Similarly, the energy sector, with its critical infrastructure increasingly connected to the internet, is a prime target for attacks that could have severe national security implications [13]. Additionally, military and governmental institutions face significant cybersecurity threats that could undermine national security, disrupt critical operations, and compromise sensitive information [20].

Cyber-attacks have become increasingly sophisticated, posing significant risks to individuals, organisations, and nations. These threats range from data breaches and ransomware to advanced persistent threats or industrial espionage. As cyber threats evolve, so too must the methods for detecting and mitigating them. Traditional security measures, such as firewalls and signature-based detection systems, are no longer sufficient to counteract the diverse and sophisticated attacks perpetrated by cybercriminals [3].

Machine learning (ML) methods offer significant advantages over traditional intrusion detection approaches by identifying complex patterns and detecting anomalies that may indicate cyber threats [2]. While classical methods rely on predefined rules and signatures, which quickly become outdated as new attack vectors emerge, ML algorithms continuously learn and adapt from vast amounts of data. This ability enables them to recognize novel and previously unknown threats that might bypass signature-based systems, a capability crucial for industries requiring timely and accurate threat detection to protect sensitive information and maintain the integrity of critical operations. ML's strength lies in its capacity to recognize patterns and anomalies in real-time, making it highly effective at identifying subtle and sophisticated attacks that traditional systems might miss. By analyzing historical data and incorporating new inputs, ML models enhance their accuracy and responsiveness over time, providing a dynamic, real-time defense mechanism against ever-evolving cyber threats [2].

In summary, the integration of ML into cybersecurity is not merely a technological advancement, but an imperative for industries confronting evolving cyber threats in today's digital era. Instead of using popular methods, we develop a novel approach where packets are stacked into windows and separately recognised. This innovative method, still relatively unexplored, offers significant potential for further

advancements in the field and represents a cutting-edge approach to enhancing cybersecurity measures.

2. Related work

The idea of monitoring and protecting computer networks is present in the literature for decades [1]. The methods for network intrusion detection systems (NIDS) have evolved along with the development of digital technologies. NIDS is a specialized subset of intrusion detection systems (IDS) that focuses on monitoring and analyzing network traffic to detect potential threats. While IDS encompasses a broader range of techniques for detecting intrusions in both network and host environments, NIDS primarily targets network-based security concerns.

Intrusion detection systems (IDS) have historically relied on a variety of techniques. Traditional methods often utilized rule-based or signature-based approaches, where predefined patterns of known attacks were used to identify malicious activities [6]. Statistical methods, anomaly detection, and expert systems have also played significant roles in the development of IDS [25]. These earlier techniques focused on recognizing deviations from normal network behaviour or comparing traffic patterns to known attack signatures. However, as the complexity of network threats increased, more adaptive and data-driven techniques emerged.

Nowadays, the authors of intrusion detection systems widely employ various ML techniques, including conventional ML methods like Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF) [8, 28] and deep learning methods e.g. Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) [12,19,24].

Talukder et al. constructed a NIDS framework [28] with integrated ML algorithms. Their approach is using efficient preprocessing, oversampling management, stacking feature embedding, and dimensionality reduction. They evaluated four ML classifiers: DT, RF, Extra Tree (ET), and Extreme Gradient Boosting (XGB) on UNSW-NB15 [16] and CIC IDS-2017 [21] datasets.

Hnamte et al. [10] explored the possibility of using deep convolutional neural networks (DCNN) in the task of network intrusion detection. They provided a comprehensive evaluation of DCNN performance in detecting various types of attacks. The evaluation was performed on publicly available IDS datasets, including ISCX-2012, DDoS (Kaggle), CIC IDS-2017, and CIC IDS-2018.

Lee et al. [12] proposed AI-based security information and event management (SIEM) system. The presented system aims to convert a large amount of security events from multiple sources like IDS, intrusion prevention systems (IPS) or firewalls (FW) to individual event profiles. Their event profiling method, designed for applying artificial intelligence techniques, provides input data with features. They evaluated proposed method on Fully Connected Neural Network (FCNN), CNN and LSTM network.

Combining a CNN as a spatial feature extractor and LSTM as a temporaly feature extractor can produce a powerful model. Praanna et al. [19] inspired by

recent advancements in computer vision tasks decided to use the CNN followed by LSTM network. The experiments on CNN-LSTM model were conducted by KDD99 dataset. Halbouni et al. [9] took the similar approach and improved it by adding dropout and batch normalization layers. The experiments were conducted by CIC IDS-2017, UNSW-NB15, and WSN-DS datasets.

Jose et al. [11] compared effectiveness of three distinct types of neural network models: deep neural network (DNN) with multiple fully connected hidden layers, LSTM, and CNN. The comparison was performed on CIC IDS-2017 dataset.

Network intrusion datasets are characterized by high class imbalance. Such datasets usually contain a lot of benign traffic. Zhang et al. [30] proposed a Parallel Cross Convolutional Neural Network (PCCN) that allows to improve the detection performance of highly imbalanced abnormal flow through feature fusion. They also improved algorithm responsible for feature extraction.

Most of the methods for NIDS [9–11,28,30] are utilising extracted traffic features (Fig. 1a). Soltani et al. [24] presented a method called Deep Intrusion Detection (DID) that can be applied to both passive and on-line traffic. The proposed method works directly on raw bytes of contents. The authors are classifying the whole frame of packets as attack or benign (Fig. 1b). The authors used a LSTM network for analysing data sequences of traffic flows. Inspired by this approach we propose a novel classification scheme that considers each packet in a window (Fig. 3). This scheme will allow us for manual verification of packets in a window and give us information about the packets which triggered the attack as there may be multiple attacks.

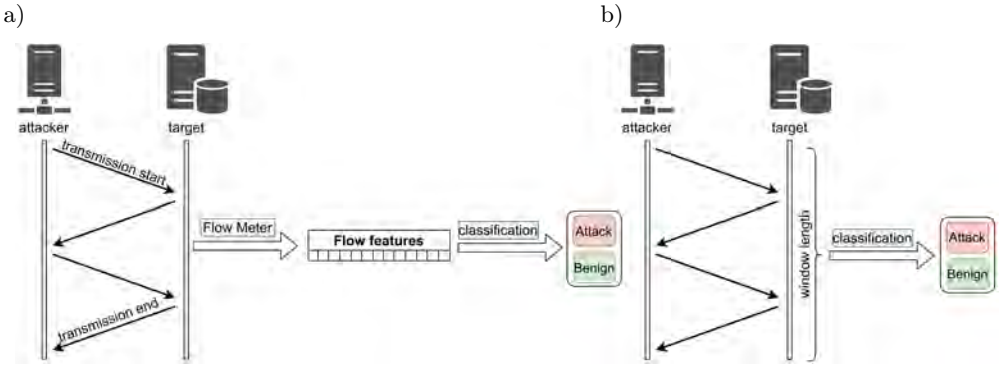


Figure 1. Approaches of classification in NIDS

Engelen et al. [7] performed a thorough methodological manual analysis of the raw CIC IDS-2017 dataset and its respective feature extraction tool named CICFlowMeter [4]. Their investigation revealed errors in attack simulation, feature extraction, labeling and benchmarking of the dataset. Based on their findings they modified the CICFlowMeter and relabeled the CIC IDS-2017 dataset. In our work we refer to modified version CICFlowMeter as improved CICFlowMeter and relabeled CIC IDS-2017 dataset as improved CIC IDS-2017 dataset.

3. Dataset

3.1. Description

The CIC IDS-2017 [21] dataset is a network intrusion dataset created by Canadian Institute for Cybersecurity (CIC) at University of New Brunswick (UNB). The dataset consists of two types of files: the PCAP files containing raw traffic data, and the CSV files, where each row corresponds to a flow including flow features and class label. The flows are constructed from raw PCAP files using the CICFlowMeter tool [14] which is also able to extract high-level statistical features like the number of packets per flow, average packet size, etc. Those features were manually engineered based on expert knowledge of traffic characteristics relevant to intrusion detection. Authors of the dataset ensured the diversity of attacks by including the most common attacks based on the 2016 McAfee report [15]. The whole dataset contains seven families of attacks: Web based, Brute force, denial-of-service (DoS), distributed denial-of-service (DDoS), Infiltration, Heartbleed and Botnet.

3.2. Preparation

As the CIC IDS-2017 and improved CIC IDS-2017 datasets contain only flow labels and our work is based on packet labels we decided to build our own data preprocessing pipeline (Fig. 2).

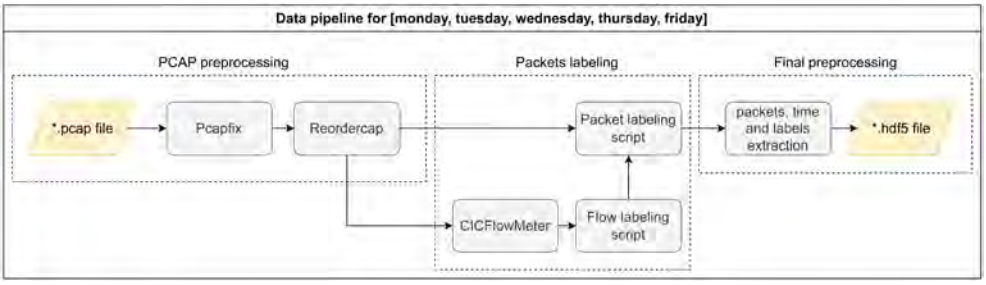


Figure 2. Data preprocessing pipeline

In the first stage of the pipeline, PCAP preprocessing is performed using Pcapfix [18] and Reordercap [14] tools if needed. Pcapfix tries to repair broken PCAP files, fixing the global header and recovering the packets by searching and guessing the packet headers. Reordercap is used to reorder and sanitize the PCAP files if needed. It ensures that the packets are in the correct chronological order.

The middle part of the pipeline is responsible for assigning labels to packets. The CICFlowMeter tool, which has been improved over the years by its original authors and others, is used for this purpose. Based on flow features generated by CICFlowMeter and the procedure provided by authors of improved CIC IDS-2017 dataset, labels are assigning to flows. The improved CICFlowMeter was extended,

which allows us to generate a CSV file with an association between the flows generated by CICFlowMeter and the packets belonging to flows that is used to assign labels to packets.

In the final stage of the pipeline all data are aggregated into HDF5 files: raw packet data, packet labels and time deltas from previous packets. The CIC IDS-2017 dataset consists of five PCAP files for five days from monday to friday. The pipeline is run for each day. Our dataset with packet labels is published at: ai.ncbj.gov.pl/datasets.

4. Methods

4.1. Solution concept

Instead of traditional flow based features, a packet based approach is adopted where at first packets are stacked into windows (subsection 4.3). Then each packet in the window is assigned to a class: *attack* – 1 or *benign* – 0 (Fig. 3).

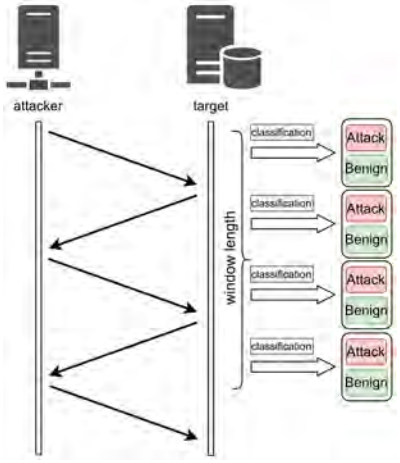


Figure 3. Assumed solution – classification of the separate packets in the packets window.

To comprehensively evaluate and compare performance, two types of ML models are employed (subsection 4.7) based on their input types:

- 1D window (single packet) input – In the context of single-packet windows, the model takes the feature vector derived from a packet and processes it through several hidden layers, with each layer performing nonlinear transformations on the input data. These transformations allow the network to learn hierarchical representations of features in a single packet.
- 2D window based models are able to capture dependencies between packets within a window. Unlike single packet analysis, where each packet is processed individually, 2D window based models consider the temporal context of packets, such as consecutive packets within a defined window.

To better explain the result of each model, its saliency map is determined (subsection 4.10).

4.2. Training

Data from the entire dataset are split into 1000 groups, which are then randomised (subsection 4.4). The order of packets in each group is maintained. This allow the original packet arrangement to be preserved as much as possible while mixing the data. The mixing of the groups for each method tests is the same, so the results are possibly comparable. The data is then split into training, validation and test sets in the proportions of 50%, 10% and 40%, respectively. This process is shown in Figure 4.

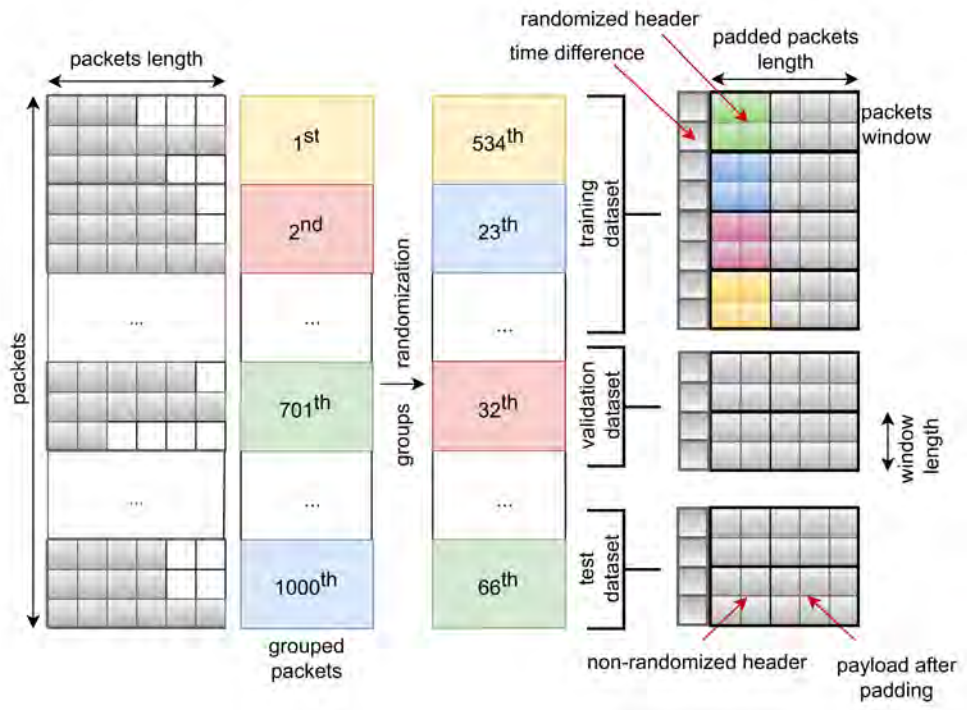


Figure 4. Packets preprocessing

Results on the validation set during training are used to select a model from each method. The model with the highest accuracy is then chosen to check the results on the test set.

4.3. Windows

Apart from the single packets, 2D packet windows are used for model learning. The shape of the window was determined mainly based on the packet length

histogram (Fig. 5). The histogram was analyzed to identify the most common packet length ranges, which led to the determination of the appropriate window width. Packets shorter than the adopted cut length were padded with zeros to ensure uniformity in the window size. Using the length of the largest packets would have resulted in excessive padding, which could negatively impact feature extraction, slow down training, and significantly increase resource consumption.

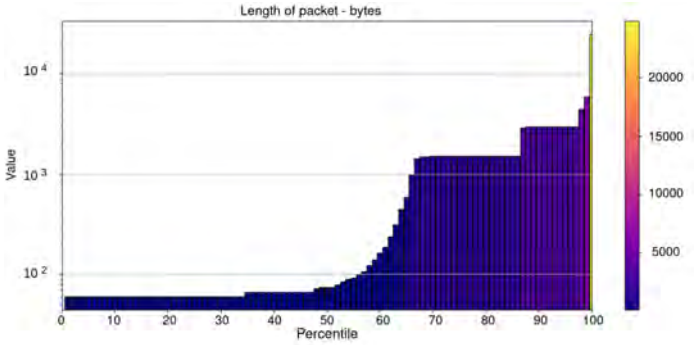


Figure 5. Histogram of packets length

The width of the windows (Fig. 6) consists of 350 bytes of packet data and one byte for the time difference between packets. This was selected experimentally, based on the histogram, to preserve the most influential features in the packets.

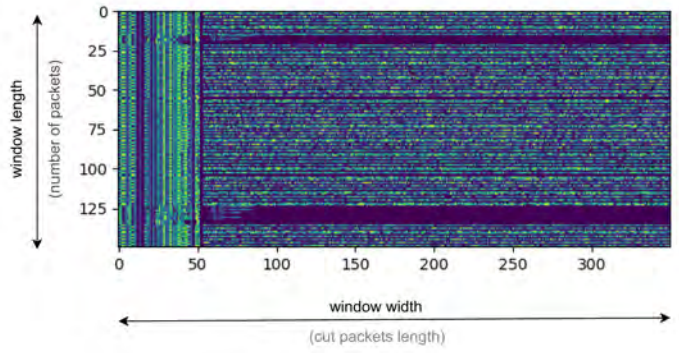


Figure 6. Example window of selected shape

Regarding the window length, which determines how many packets are included, the choice was influenced by both accuracy and computational constraints. While selecting a window length slightly longer than 1000 packets seemed optimal for capturing more information, it would have significantly increased resource consumption and slowed down training. Therefore, the final value of 150 packets per window was

selected experimentally, balancing the need for sufficient data in each window and the desire to minimize computational overhead.

In every window two parts can be distinguished (Fig. 6). The first structured part, usually up to about 54 bytes (TCP protocol), is the header. The rest of the windows is payload of the packets.

4.4. Randomised replacement

Packets randomisation is perfomed to prevent the model from focusing on the specific data like MAC or IP address. Most of the other solutions [24] that have been found, assume cutting out these particular parts of the packet header. However, removing this data removes some inter-packet dependencies when packets are stacked into a 2D window. Therefore, every packet has a randomised destination and source MAC address. IP addresses and ports are also randomised when they occur. After randomisation, new checksums are calculated. Randomisation is done in such a way that, for example, if the value of a port is changed from one to another, all destination and source ports of that primary value are changed consistently to maintain dependencies. Therefore, it can be called as *randomised replacement*.

Figure 7 shows the first 70 bytes of a one package window before and after randomisation. Black colour means value 0, white 255.

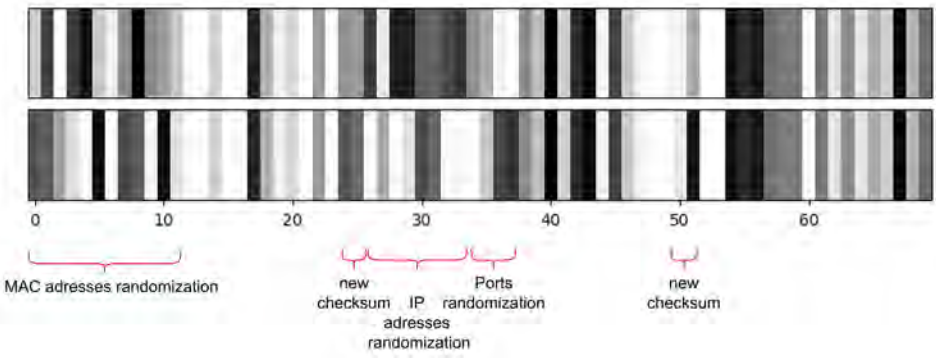


Figure 7. Example header randomisation

4.5. Labeling

One of the advantages of having each packet labelled separately (Fig. 3), is the ability to test two types of packet labeling. The first where only movement from attacker (input traffic) is labelled as an attack (Fig. 8a)) and the second where response from target to attacker (output traffic) is also treated as a threat (Fig. 8b)).

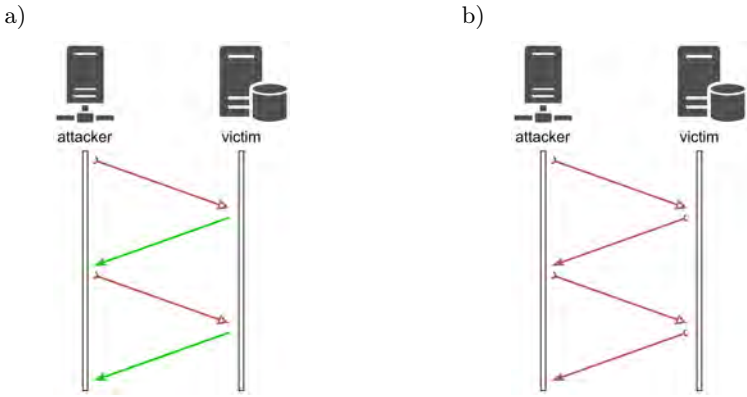


Figure 8. Two types of attacks labeling

4.6. Oversampling

Due to the unbalanced dataset, two cases for each model are considered: training on the imbalanced dataset and training with minority class oversampling on the training dataset.

The oversampling is performed to obtain the same number of benign and attack packets, in case of single packet input models, and the same number of windows containing at least one infected packet as number of fully benign windows for the 2D input methods.

4.7. Models

4.7.1. Fully connected neural network

The fully connected neural network (Fig. 9) processes each packet individually, so is unable to capture dependencies between successive packets. The result is based on feature dependencies within each individual packets. The model comprises three fully connected layers, consisting of 256, 356, and 32 neurons. Two of these layers are followed by batch normalisation and dropout layers.

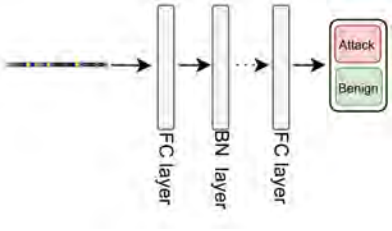


Figure 9. Fully connected neural network (FCNN) simplified architecture

Model characteristic and hyperparameters:

- input: 1 packet × 351 bytes
- output: 1 binary classification,
- optimiser: Adam,
- initial learning rate: 0.001,
- batch size: 8096.

4.7.2. Convolutional neural network

The convolutional neural network (Fig. 10) with a three convolutional filters followed by max-pooling layers was tested. The filters are larger than commonly used shapes [23, 27], such as 9×9 and 7×7. As smaller filters resulted in worse accuracy. Feature maps are followed by 2×2 max pooling layers.

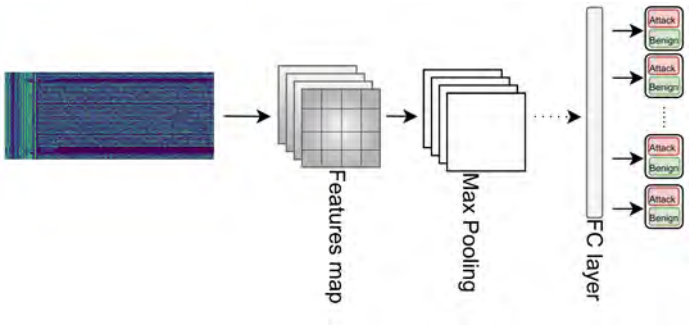


Figure 10. Convolutional neural network (CNN) simplified architecture

Model characteristic and hyperparameters:

- input: 150 packets × 351 bytes,
- output: 150 binary classifications,
- optimiser: Adam,
- initial learning rate: 0.001,
- batch size: 64.

4.7.3. Hybrid neural network

The hybrid neural network (Fig. 11) consists of a 1D convolutional operations with six filters and 1×3 window shape, that extracts spatial patterns within each packet separately. LSTM layers are then adapted to process the sequential information extracted from the convolutional layers. The architecture with 1D CNN performs better than LSTM preceded by 2D CNN and LSTM only model

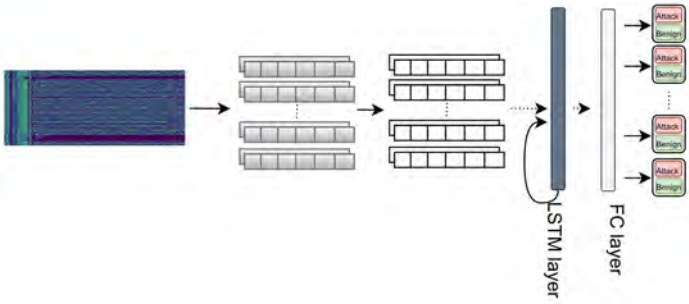


Figure 11. Hybrid neural network (CNN-LSTM) simplified architecture

Model characteristic and hyperparameters:

- input: 150 packets × 351 bytes,
- output: 150 binary classifications,
- optimiser: Adam,
- initial learning rate: 0.0005,
- batch size: 64.

4.7.4. EfficientNet-based neural network

Architecture based on EfficientNet B0 [29], preceded by a convolutional layer to match the required RGB format (Fig. 12). Model comprise pretrained *Imagenet* [5] weights. Model is fine-tuned on new data.

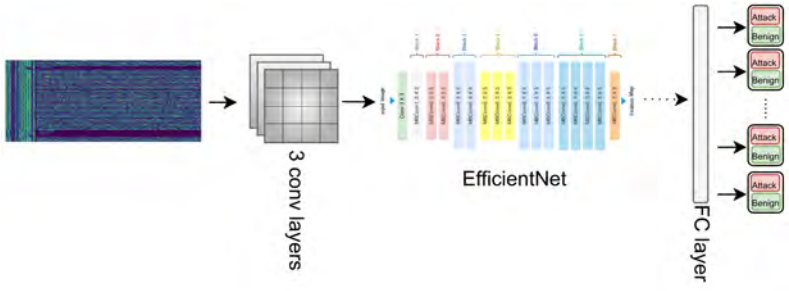


Figure 12. EfficientNet based neural network simplified architecture

Model characteristic and hyperparameters:

- input: 150 packets × 351 bytes,
- output: 150 binary classifications,
- optimiser: Adam,
- initial learning rate: 0.0005,
- batch size: 16.

4.8. Metrics

The following metrics are used for evaluation of classification tasks:

- **Accuracy**: Measures the proportion of correctly classified instances among all instances.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total number of packets}}$$

- **Precision**: Measures the proportion of correctly predicted positive instances among all instances predicted as positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall**: Measures the proportion of correctly predicted positive instances among all actual positive instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Accuracy, despite its simplicity and ease of comparison with other solutions, may provide a misleading picture due to class imbalance. Therefore, while accuracy is useful for general comparison purposes with other solutions, precision and recall offer more nuanced insights into model performance. Due to the imbalance, the accuracy is skewed by the most prevalent benign class.

4.9. Loss functions

Different loss functions were tested for our task: Binary Crossentropy, Focal Loss, Dice Loss, and Intersection over Union.

- **Binary Crossentropy:**

$$\text{Binary Crossentropy} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right] \tag{1}$$

where:

N – is the number of samples in window,

y_i – is the true label,

p_i – is the predicted probability.

- **Focal Loss:**

$$\text{Focal Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[\alpha(1 - p_i)^{\gamma} y_i \log(p_i) + (1 - \alpha)p_i^{\gamma}(1 - y_i)\log(1 - p_i)\right] \tag{2}$$

where:

α is weighting factor used to deal with class imbalance,

γ is focusing parameter. Increasing the value increases the sensitivity to misclassified observations.

- **Dice Loss:**

$$\text{Dice Loss} = 1 - \frac{2\sum_i^N y_i p_i}{\sum_i^N y_i + \sum_i^N p_i} \tag{3}$$

- **Intersection over Union (IoU):**

$$\text{IoU} = 1 - \frac{\sum_i^N y_i p_i}{\sum_i^N y_i + \sum_i^N p_i - \sum_i^N y_i p_i} \tag{4}$$

Experimentes with Dice Loss and IoU, treating the problem as a specific segmentation task, yielded very poor results.

After evaluating the performance of each cost function, Binary Crossentropy provided the best results for our task.

4.10. Saliency maps

Saliency maps are a crucial tool in the field of deep learning and computer vision, particularly for understanding and interpreting the decisions made by neural networks.

One common method to compute a saliency map is via gradient backpropagation. The idea is to compute the gradient of the highest output score with respect to the input image [22]. In this case the image is represented as a window of packets. This gradient indicates how much a change in each pixel value would affect the highest output score. Unlike in image classification, we are not interested in a particular area of a particular image, so the results are averaged over the entire batch. Mathematically, this is represented as:

$$S_{ij} = \left| \frac{1}{B} \sum_{k=1}^{B} \frac{\partial \max(\mathbf{y}^{(k)})}{\partial x_{ij}^{(k)}} \right|$$

where:

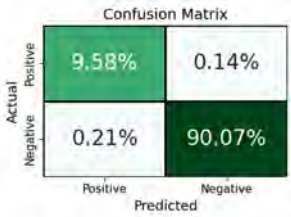
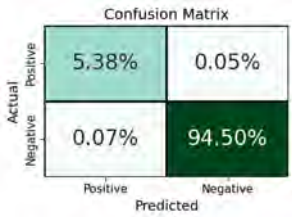
S_{ij} – is the saliency value for the data at position (i, j),
$\mathbf{y}^{(k)}$ – is the vector of output scores predicted by the neural network for all classes for the k-th window in the batch,
$x_{ij}^{(k)}$ – is the pixel value at position (i, j) in the input image $\mathbf{X}^{(k)}$, which is the k-th window in the batch,
B – is the batch size.

5. Results and discussion

5.1. Fully connected neural network

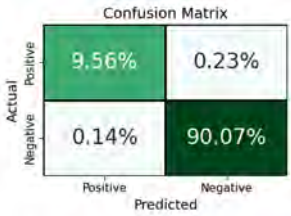
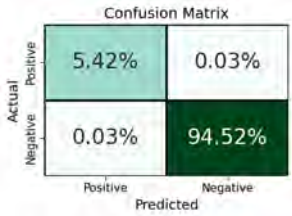
Results obtained on the test dataset from FCNN in the form of confusion matrices from two labelling methods are shown in Table 1 with the imbalanced dataset and in Table 2 with attack packets oversampling.

Table 1
Results obtained from FCNN with imbalanced train set

Both side labeling	Forward attack labeling
	

Not marking backward route as attacks works better – the network needs to find less features and also responses taken out of context should be similar to normal network traffic. The confusion matrix generated from the best case is shown. A 0.03% false negatives rate is obtained, which is the most important characteristic to minimise, as it means missing an incoming attack.

Table 2
Results obtained from FCNN with balanced train set

Both side labelling	Forward attack labelling
	

Saliency map obtained from the batch from the test dataset (Fig. 13) shows that results are strongly dependent on the packet header. Header features are strongly dependent on network characteristic such as topology, size, or type. The most influential byte correspond to the Time to Live value in IP protocol, which indicates the validity period of packet data.

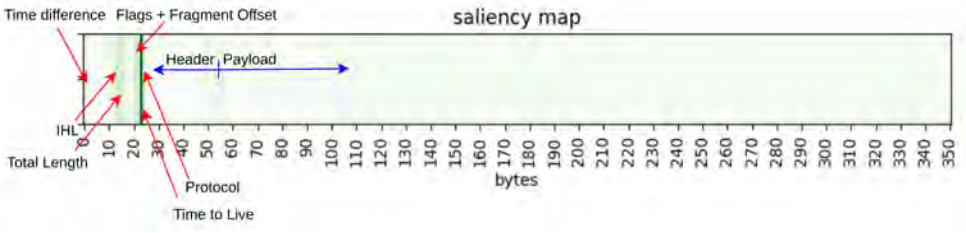


Figure 13. Saliency map obtained from the best FCNN model

5.2. Convolutional neural network

In case of the CNN, class balancing does not visibly improve the results (Tab. 3) – they are similar to the results with an imbalanced dataset (Tab. 4). Unlike in single-packet input models (FCNN), not tagging return packets as attacks (Fig. 8) in 2D window input models creates difficulties in distinguishing patterns or features between packets within the window. This happens because the model processes information from consecutive packets, making it harder to differentiate between normal and attack traffic. As a result, performance degrades, and the model struggles to identify attacks in reverse traffic. That effect occurs in all window based models.

Table 3
Results on the test set obtained from CNN with imbalanced train set

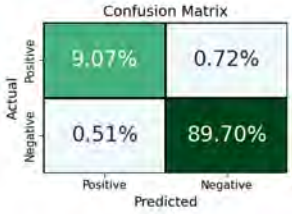
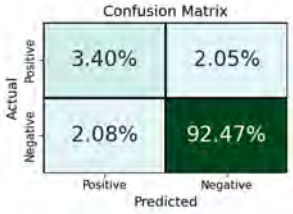
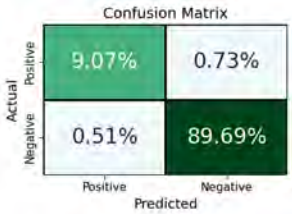
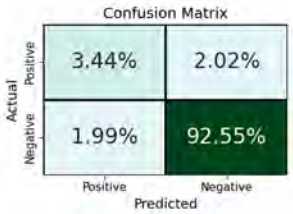
Both side labelling	Forward attack labelling
	

Table 4
Results on the test data obtained from CNN with balanced train set

Both side labelling	Forward attack labelling
	

The Metric values in this case are worse than in FCNN; however, the saliency map shows that features are found in both the header and the payload (Fig. 14). This may result in a better ability to generalize on other datasets because the model learns to capture content-based features from the payload, which are more independent of the specific network. In contrast, the header contains more network-specific features, and focusing on it might limit the model's ability to generalize. By learning from both the header and payload, the model can achieve a better balance, making it more adaptable to different types of network traffic. Large convolutional filters cause to find large areas of interest that are visible on the packets payload part of the saliency map (Fig. 14).

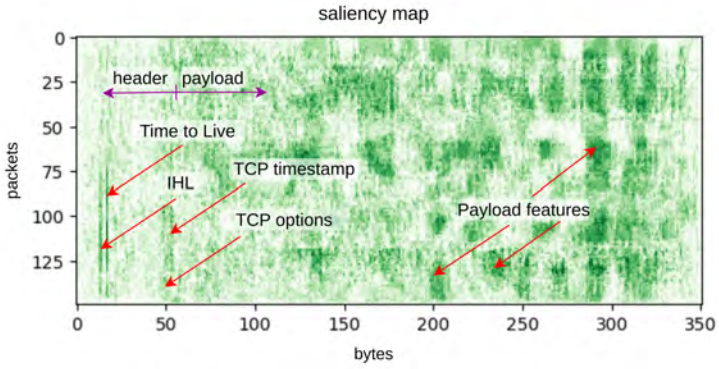


Figure 14. Saliency map obtained from the best CNN model

5.3. Hybrid neural network

Confusion matrices obtained from hybrid neural network (CNN-LSTM) for two labelling methods are shown in Table 6. This shows that class balancing has improved the results. The model is unable to generalise and detect attacks on the validation and test sets when responses are not marked as an attack (Tab. 5). The LSTM part seems to be more sensitive to imbalanced data than the other ones used.

Table 5

Results on the test set obtained from hybrid neural network with imbalanced train set

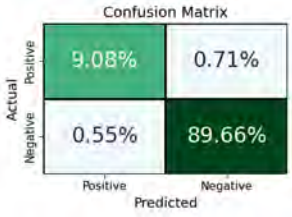
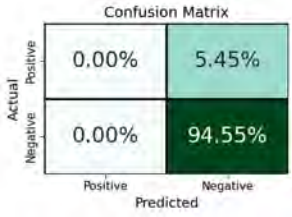
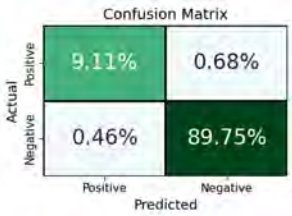
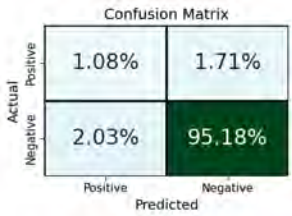
Both side labelling	Forward attack labelling
	

Table 6

Results on the test data obtained from hybrid neural network with balanced train set

Both side labelling	Forward attack labelling
	

Saliency map (Fig. 15) shows that the model takes features similarly from header and payload. The model is able to consider features between packets, which is visible as vertical lines. The model can also detect some interesting packets sequences, indicated by horizontal lines.

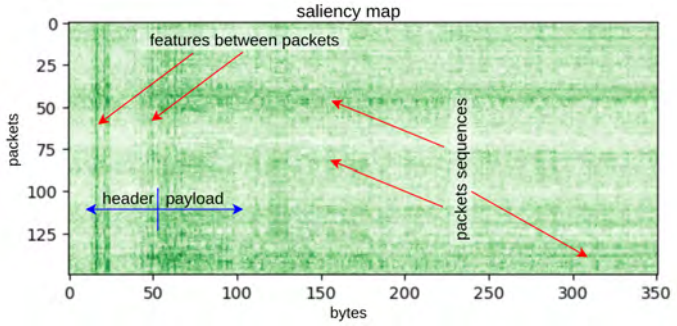


Figure 15. Saliency map obtained from the best hybrid neural network model

5.4. EfficientNet based neural network

EfficientNet based neural network with pretrained Imagenet weights gives the best result from window input based models for balanced (Tab. 7) and unbalanced data (Tab. 8). At the same time, it is much more complex and slower.

Table 7

Results on the test set obtained from EfficientNet based neural network with imbalanced data

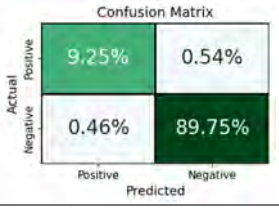
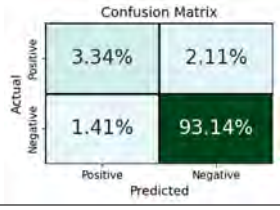
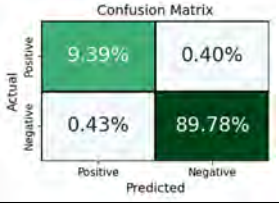
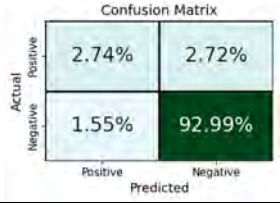
Both side labelling	Forward attack labelling
	

Table 8

Results on the test set obtained from EfficientNet based neural network with balanced train set

Both side labelling	Forward attack labelling
	

In addition, saliency map (Fig. 16) is laid out uniformly without paying attention to any individual elements, which may lead to the best generalisation ability.

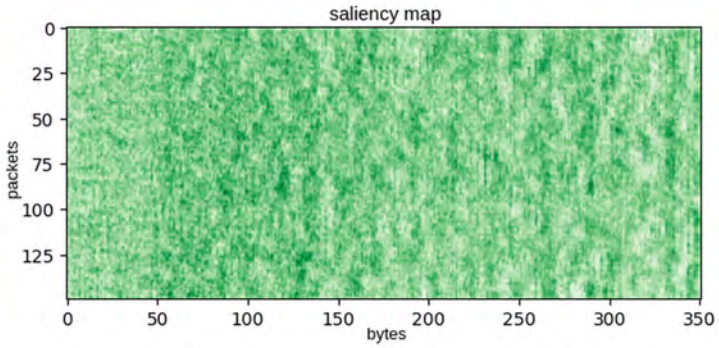


Figure 16. Saliency map obtained from the best EfficientNet based neural network model.

5.5. Comparison

The accuracy of our result obtained from FCNN model (Tab. 9 – green) is better or comparable to most of the methods based on flow features. Most of the highest-value results are based on random forests, which excel at selecting and combining features from packets flow characteristics. However, they would not perform well in packets window based methods due to their struggle with high dimensionality data, the inability to capture spatial correlations, and difficulty in modelling complex patterns. Compared to the flow based methods the obtained recall and precision values are lower, which is expected when considering packets individually in respect to full flows.

On the other hand, the results are comparable to those obtained with the *DID (LSTM)* algorithm (Tab. 9 – bolded), noting that when labelling single packets, the metrics will tend to reach lower values than when classifying the full window. The results for window based methods are sub par (Tab. 9 – purple) when compared to single packet solution or flow features based models. However the analysis of saliency maps shows that those models should be more robust with other data, which requires further investigation.

Table 9
Related works on intrusion detection using CIC-IDS-2017 dataset

Method	Accuracy [%]	Recall [%]	Precision [%]	Input Type	Classification (of)
RF [7]	99.99	99.99	99.99	Flow features	Flow
DCNN [10]	99.96	99.96	99.96	Flow features	Flow
ET [28]	99.95	99.95	99.95	Flow features	Flow
RF [28]	99.94	99.94	99.94	Flow features	Flow
DT [28]	99.91	99.91	99.91	Flow features	Flow

Table 9 cont.

Method	Accuracy [%]	Recall [%]	Precision [%]	Input Type	Classification (of)
CNN [11]	99.61	95.00	97.05	Flow features	Flow
XGB [28]	99.65	99.65	99.65	Flow features	Flow
CNN-LSTM [9]	99.48	99.69	99.25	Flow features	Flow
EP-FCNN [12]	99.50	–	–	Flow features	Flow
CNN-LSTM [19]	99.78	–	–	Flow features	Flow
CNN [19]	99.23	–	–	Flow features	Flow
EP-CNN [12]	98.80	–	–	Flow features	Flow
DT [8]	98.80	97.30	–	Flow features	Flow
EP-LSTM [12]	98.60	–	–	Flow features	Flow
DBN [19]	98.59	–	–	Flow features	Flow
SVM [19]	98.20	–	–	Flow features	Flow
LSTM [11]	97.67	95.95	94.96	Flow features	Flow
DNN [11]	90.61	84.60	80.85	Flow features	Flow
DID (LSTM) [24]	–	**99.80**	**99.20**	**Window**	**Window**
FCNN	**99.93**	**99.41**	**99.37**	**Window**	**Packets**
CNN	**98.77**	**94.66**	**92.65**	**Window**	**Packets**
CNN+LSTM	**98.85**	**95.18**	**93.01**	**Window**	**Packets**
Eff-Net based	**99.17**	**95.61**	**95.88**	**Window**	**Packets**

6. Conclusions and overlook

6.1. Summary

The four proposed models demonstrate the ability to classify individual packets, proving that replacing flow-features based models with operations directly on packets is possible. The most important conclusion is that while FCNN has the best theoretical metric values, it is strongly dependent on packets headers. This highlights that explaining the results should be one of the most important aspects of cybersecurity ML approaches.

The 2D input based model can result in better generalisation ability on other datasets. Window based models can also work well as initial weights to adapt the model to other dataset or to perform fine-tuning.

A potentially interesting approach would be the creation of an ensemble model that combines fully connected neural networks (FCNN) with 2D input-based models. By weighted averaging the predictions from these different models, the ensemble could leverage the strengths of each approach, potentially leading to improved overall performance.

6.2. Future plans

The quickest way to potentially improve performance would be to test windows with a significantly larger number of bytes by checking the cutoff at the 85th and 97th

percentiles of packet lengths (5). Larger windows would likely capture more detailed information, which could help identify subtle attack patterns. However, these larger windows would also impose memory constraints on the GPU, requiring a reduction in batch size to fit within available memory. This would significantly increase training time unless more powerful hardware is available to handle the additional data more efficiently. Other intriguing possibilities inlcude: dynamic windows shape where the width of the window depends on the largest packet in each window, and dynamic window length where number of packets could depend on the selected time value.

An EfficientNet based model demonstrates a significant promise by leveraging both pretrained weights and a window based approach, which supports comprehensive learning of various features and patterns throughout packets window. The model's potential is further validated by saliency map analyses, which highlight its capacity for effective generalization in various datasets. To fully assess and capitalize on this potential, it is essential to test the model on a variety of cybersecurity datasets, such as KDD Cup 1999, and UNSW-NB15, as well as more domain-specific datasets.

Acknowledgments

Work done as part of the CYBERSECIDENT/369195/I/NCBR/2017 project supported by the National Centre of Research and Development in the frame of CyberSecIdentProgramme.

References

[1] Anderson J.P.: Computer security threat monitoring and surveillance, *Technical Report, James P Anderson Company*, 1980.

[2] Buczak A., Guven E.: A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, *IEEE Communications Surveys & Tutorials*, vol. 18(2), pp. 1153–1176, 2016. doi: 10.1109/COMST.2015.2494502.

[3] Chen P., Desmet L., Huygens C.: A Study on Advanced Persistent Threats. In: B. De Decker, A. Zúquete (eds.), *Communications and Multimedia Security. CMS 2014*, Lecture Notes in Computer Science, vol. 8735, pp. 63–72, Springer, Berlin, Heidelberg, 2014. doi: 10.1007/978-3-662-44885-4_5.

[4] CICFlowMeter tool, https://www.unb.ca/cic/research/applications.html. Accessed: 2024-05-05.

[5] Deng J., Dong W., Socher R., Li L.J., Li K., Fei-Fei L.: ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[6] Díaz-Verdejo J., Muñoz Calle J., Estepa Alonso A., Estepa Alonso R., Madinabeitia G.: On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks, *Applied Sciences*, vol. 12(2), 2022. doi: 10.3390/app12020852.

[7] Engelen G., Rimmer V., Joosen W.: Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In: *2021 IEEE Security and Privacy Workshops (SPW)*, pp. 7–12, IEEE, 2021. doi: 10.1109/spw53761.2021.00009.

[8] Guezzaz A., Benkirane S., Azrour M., Khurram S.: A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality, *Security and Communication Networks*, vol. 2021(1), 1230593, 2021. doi: 10.1155/2021/1230593.

[9] Halbouni A., Gunawan T.S., Habaebi M.H., Halbouni M., Kartiwi M., Ahmad R.: CNN-LSTM: hybrid deep neural network for network intrusion detection system, *IEEE Access*, vol. 10, pp. 99837–99849, 2022. doi: 10.1109/access.2022.3206425.

[10] Hnamte V., Hussain J.: Dependable intrusion detection system using deep convolutional neural network: A novel framework and performance evaluation approach, *Telematics and Informatics Reports*, vol. 11, 2023. doi: 10.1016/j.teler.2023.100077.

[11] Jose J., Jose D.V.: Deep learning algorithms for intrusion detection systems in internet of things using CIC-IDS 2017 dataset, *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13(1), pp. 1134–1141, 2023. doi: 10.11591/ijece.v13i1.pp1134-1141.

[12] Lee J., Kim J., Kim I., Han K.: Cyber threat detection based on artificial neural networks using event profiles, *IEEE Access*, vol. 7, pp. 165607–165626, 2019. doi: 10.1109/access.2019.2953095.

[13] Makrakis G.M., Kolias C., Kambourakis G., Rieger C., Benjamin J.: Industrial and Critical Infrastructure Security: Technical Analysis of Real-Life Security Incidents, *IEEE Access*, vol. 9, pp. 165295–165325, 2021. doi: 10.1109/ACCESS.2021.3133348.

[14] Mathieson M.: Reordercap tool. https://www.wireshark.org/docs/man-pages/reordercap.html. Accessed: 2024-05-05.

[15] McAfee report, 2016. https://web.archive.org/web/20171026083736/https://www.mcafee.com/us/resources/reports/rp-threats-predictions-2016.pdf. Accessed: 2024-05-05.

[16] Moustafa R., Slay J.: A comprehensive data set for network intrusion detection systems, *School of Engineering and Information Technology University of New South Wales at the Australian Defense Force Academy Canberra, Australia, UNSW-NB15*, 2015.

[17] Muthuppalaniappan Menaka L., Stevenson K.: Healthcare cyber-attacks and the COVID-19 pandemic: an urgent threat to global health, *International Journal for Quality in Health Care*, vol. 33(1), mzaa117, 2020. doi: 10.1093/intqhc/mzaa117.

[18] Pcapfix. https://github.com/Rup0rt/pcapfix. Accessed: 2024-05-05.

[19] Praanna K., Sruthi S., Kalyani K., Tejaswi A.S.: A CNN-LSTM model for intrusion detection system from high dimensional data, *Journal of Information and Computational Science*, vol. 10(3), pp. 1362–1370, 2020. doi: 10.5281/zenodo.7911821.

[20] Rid T., Buchanan B.: Attributing Cyber Attacks, *Journal of Strategic Studies*, vol. 38(1-2), pp. 4–37, 2015. doi: 10.1080/01402390.2014.977382.

[21] Sharafaldin I., Lashkari A.H., Ghorbani A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy ICISSP – Volume 1*, pp. 108–116, 2018. doi: 10.5220/0006639801080116.

[22] Simonyan K., Vedaldi A., Zisserman A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 2014. https://arxiv.org/abs/1312.6034.

[23] Simonyan K., Zisserman A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, *arXiv preprint arXiv:14091556*, 2014.

[24] Soltani M., Siavoshani M.J., Jahangir A.H.: A content-based deep intrusion detection system, *International Journal of Information Security*, vol. 21(3), pp. 547–562, 2022.

[25] Sulaiman N.S., Nasir A., Othman W., Fahmy S., Aziz N., Yacob A., Samsudin N.: Intrusion Detection System Techniques: A Review, *Journal of Physics: Conference Series*, vol. 1874, 012042, 2021. doi: 10.1088/1742-6596/1874/1/012042.

[26] Symantec Corporation: Internet Security Threat Report, *Symantec Corporation*, 2017.

[27] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z.: Rethinking the Inception Architecture for Computer Vision. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

[28] Talukder M.A., Islam M.M., Uddin M.A., Hasan K.F., Sharmin S., Alyami S.A., Moni M.A.: Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction, *Journal of Big Data*, vol. 11(1), p. 33, 2024. doi: 10.1186/s40537-024-00886-w.

[29] Tan M., Le Q.V.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2020.

[30] Zhang Y., Chen X., Guo D., Song M., Teng Y., Wang X.: PCCN: parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows, *IEEE Access*, vol. 7, pp. 119904–119916, 2019. doi: 10.1109/access.2019.2933165.

Affiliations

Aleksander Ogonowski
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7,
Aleksander.Ogonowski@ncbj.gov.pl

Michał Żebrowski
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7,
Michal.Zebrowski@ncbj.gov.pl

Arkadiusz Ćwiek
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Soltana 7, Arkadiusz.Cwiek@ncbj.gov.pl

Tobiasz Jarosiewicz
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7, Tobiasz.Jarosiewicz@ncbj.gov.pl

Konrad Klimaszewski
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7, Konrad.Klimaszewski@ncbj.gov.pl

Adam Padee
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7, Adam.Padee@ncbj.gov.pl

Piotr Wasiuk
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7, Piotr.Wasiuk@ncbj.gov.pl

Michał Wójcik
National Centre For Nuclear Research, 05-400 Otwock-Świerk ul. A. Sołtana 7, Michal.Wojcik@ncbj.gov.pl

Received: 11.03.2025
Revised: 12.03.2025
Accepted: 12.03.2025