I.J. Wireless and Microwave Technologies, **2025, 4, 65-72**
Published Online on August 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijwmt.2025.04.05

The Impact of Artificial Intelligence on Cybersecurity and Data Protection

Iryna Zavushchak*
Institute of Computer Sciences and Information Technologies, Department of Information Systems and Networks.
Lviv Polytechnic National University, Lviv, Ukraine
E-mail: iryna.i.zavushchak@lpnu.ua
ORCID iD: https://orcid.org/ 0000-0002-5371-8775
*Corresponding Author

Received: 25 October, 2024; Revised: 27 December, 2024; Accepted: 20 February, 2025; Published: 08 August, 2025

Abstract: The escalating complexity of cybersecurity threats necessitates advanced technological solutions to protect digital infrastructures. This study explores the application of Autoencoder neural networks, a deep learning model, for anomaly detection in network traffic, aiming to enhance real-time identification of cyberattacks. Using the CICIDS2017 dataset, which encompasses diverse attack types such as Distributed Denial of Service (DDoS) and infiltration, the Autoencoder was trained to detect deviations from normal traffic patterns based on reconstruction errors. The model was optimized through preprocessing, feature selection, and hyperparameter tuning, achieving strong performance metrics including precision, recall, F1-score, accuracy, and ROC-AUC. Despite its effectiveness in distinguishing normal and malicious traffic, challenges arose in detecting stealthy attacks like slow brute-force attempts. These results underscore the Autoencoder's potential in cybersecurity frameworks and highlight opportunities for improvement through adaptive thresholds and hybrid models. This study contributes to advancing AI-driven anomaly detection, promoting proactive defense against evolving cyber threats.

Index Terms: Artificial Intelligence, Cybersecurity, Anomaly Detection, Autoencoder, Network Traffic Analysis, Machine Learning, Intrusion Detection, Data Protection, Deep Learning, Threat Detection

1. Introduction

The rapid growth of digital infrastructures has led to an increase in the complexity and frequency of cyberattacks, posing significant challenges for organizations striving to protect sensitive data and ensure system reliability. Traditional security solutions, such as firewalls and signature-based Intrusion Detection Systems (IDS), are no longer sufficient to address sophisticated and evolving cyber threats, including zero-day attacks. These conventional methods rely on predefined attack patterns, making them ineffective against previously unseen threats or subtle, stealthy activities. As a result, there is a growing need for more dynamic and proactive cybersecurity solutions.

Artificial Intelligence (AI) has emerged as a powerful tool in the field of cybersecurity, offering the ability to analyze large volumes of data and detect anomalies that may indicate malicious activities. Among various AI techniques, machine learning models have gained prominence for their ability to identify both known and unknown threats. One particularly promising approach is the use of Autoencoder neural networks, an unsupervised deep learning model capable of learning normal behavior patterns and detecting deviations without requiring labeled datasets. This makes Autoencoders ideal for cybersecurity applications, where labeling data is often challenging and costly[1].

The specific objective of this study is to evaluate the effectiveness of an Autoencoder neural network in detecting anomalies within network traffic. The research leverages the CICIDS2017 dataset, a comprehensive set of network data that includes both benign and malicious traffic, simulating real-world attack scenarios. This paper aims to answer the following research questions:

How well can an Autoencoder detect anomalies in network traffic compared to traditional methods?

What are the optimal hyperparameters and thresholds for achieving a balance between false positives and true positives?

What challenges and limitations arise when applying Autoencoders in dynamic network environments?

This study contributes to the growing body of knowledge on AI-based cybersecurity solutions by exploring the potential of Autoencoders for real-time threat detection. The findings are expected to provide insights into how Autoencoders can enhance anomaly detection capabilities, identify zero-day attacks, and integrate with broader cybersecurity frameworks.

2. Materials and Methods

To evaluate the effectiveness of the Autoencoder in detecting anomalies in network traffic, a comprehensive study was conducted using real-world data and state-of-the-art machine learning techniques[2]. The primary objective of the experimental part was to develop a model capable of learning from normal traffic and accurately detecting deviations that indicate potential attacks. The selection of an appropriate dataset, proper data preprocessing, and careful tuning of the model's architecture were critical steps that impacted the final outcomes.

For this study, CICIDS2017 was chosen—one of the most widely used datasets for intrusion detection research. The dataset includes both normal traffic and multiple types of attacks, allowing us to test the model's ability not only to detect known threats but also to identify new, previously unseen patterns. The capability to automatically detect anomalies without predefined labels is particularly valuable in dynamic network environments, where new threats can emerge unpredictably[3,4].

This section outlines the necessary steps, including data selection, preprocessing, the architecture of the Autoencoder model, and the methods used for training and testing. Special attention is given to the tuning of threshold values for anomaly detection, as these thresholds significantly influence the trade-off between false positives and false negatives.

2.1 Dataset Overview: CICIDS2017

The CICIDS2017 dataset, developed by the Canadian Institute for Cybersecurity[6], simulates real-world network traffic, including both normal behavior and various cyberattacks. This dataset provides a comprehensive environment to test and benchmark intrusion detection systems. It contains over 3 million records with 78 features per record, such as packet size, protocol type, flow duration, and TCP flags. The data is well-structured to capture both high-level network behavior and low-level packet details.

The dataset was collected over multiple days, with each day representing a different attack scenario. For instance, Monday's traffic consists entirely of normal behavior, serving as a baseline for training anomaly detection models. Other days include attacks such as:

- Brute-force attacks: Repeated login attempts to gain unauthorized access.
- DDoS attacks: Flooding a network or server with traffic to make it unavailable.
- Infiltration and botnet activity: Unauthorized system access followed by malicious operations.

This combination of attack types, along with benign traffic, allows for thorough testing of AI models. One challenge with the CICIDS2017 dataset is class imbalance, where certain attack types have fewer instances than others, which requires careful handling during preprocessing to prevent biased model performance. Additionally, feature normalization is necessary to ensure that features with different scales do not negatively affect the model's training process.

Overall, CICIDS2017 provides an ideal platform for evaluating anomaly detection methods like Autoencoders, as it offers labeled traffic data and a realistic mix of normal and malicious network activities.

2.2 Data Preprocessing

Effective data preprocessing ensures that the Autoencoder can accurately learn the patterns in normal traffic and identify anomalies. Several steps were applied to prepare the CICIDS2017 dataset for training and evaluation:

1. Handling Missing Values:

Some records in the dataset contained missing or incomplete values[5]. These were replaced either with zero (for numerical features) or the mean value of the feature across the dataset to maintain consistency.

2. Feature Selection:

To prevent data leakage and improve training efficiency, irrelevant features such as Flow ID, IP addresses, and Timestamps were removed. These fields are not indicative of network behavior and could bias the model.

3. Encoding Labels:

Labels were used only for evaluation purposes, as the Autoencoder is an unsupervised model. The "BENIGN" label indicates normal traffic, while other labels represent different attack types. These were kept in the test set to assess the model's ability to identify anomalies.

4. Normalization:

The features in network traffic data vary widely in scale (e.g., packet size vs. flow duration). StandardScaler was applied to normalize all features, ensuring that large values do not dominate the training process.

5. Train-Test Split:

Training set: Contained only normal traffic from Monday's data to allow the Autoencoder to learn patterns of benign activity.

Test set: Included both normal and attack traffic from other days, ensuring the model could generalize and detect anomalies effectively.

2.3 Autoencoder Model Architecture

The Autoencoder model used in this study is designed to learn a compact representation of normal network behavior. It consists of two main components:

1. Encoder:

Reduces the dimensionality of input data and extracts key features.
Uses Dense layers with ReLU activation to introduce non-linearity.

2. Latent Space:

The core representation where compressed data is stored.
This space holds essential patterns learned from normal traffic, with fewer dimensions than the input.

3. Decoder:

Attempts to reconstruct the original input from the latent space.
A sigmoid activation function is used in the output layer to constrain values between 0 and 1.
The complete Autoencoder architecture is implemented as follows(Fig.1):

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

autoencoder = Sequential([
    Dense(64, activation='relu', input_shape=(78,)),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),  # Latent space
    Dense(32, activation='relu'),
    Dense(64, activation='relu'),
    Dense(78, activation='sigmoid')  # Output layer
])

autoencoder.compile(optimizer='adam', loss='mse')
```

Fig. 1. Example of realization Autoencoder architecture

Loss Function: The Mean Squared Error (MSE) measures the reconstruction error between the input and output, indicating whether the input is similar to learned normal patterns.

Optimizer: Adam was selected for its ability to converge quickly during training.

2.4 Experimental Setup and Training

The Autoencoder was trained on normal traffic to learn the typical patterns present in benign flows. Several important hyperparameters were configured to ensure the model's effectiveness:

1. Training Parameters:

Epochs: 50 (the number of complete passes through the training data).
Batch Size: 32 (the number of samples processed before updating the model's weights).

2. Validation Split:

During training, 10% of the data was reserved for validation[7]. This ensured that the model generalizes well without overfitting to the training data.

3. Threshold Calculation:

After training, the reconstruction error for each sample in the test set was calculated. A threshold for anomaly detection was set based on the distribution of these errors(1):

$$Threshold = \mu + 2\sigma \tag{1}$$

where μ is the mean reconstruction error and σ is the standard deviation. If the error for a given sample exceeds the threshold, the sample is classified as anomalous.

4. Performance Evaluation:

The model's performance was tested on the labeled test set, containing both normal and attack traffic. Key metrics such as Precision, Recall, F1-score, and ROC-AUC were used to assess the detection capabilities.

This experimental setup ensures that the Autoencoder can accurately reconstruct normal traffic while identifying deviations indicative of malicious activities. The use of a validation set helps prevent overfitting, ensuring that the model performs well on unseen data.

3. Results

This section presents the performance of the Autoencoder model for anomaly detection, with a focus on the experimental findings, analysis of metrics, and interpretation of the results. The trained Autoencoder was evaluated on a test set containing both normal traffic and various cyberattacks from the CICIDS2017 dataset. This analysis covers the impact of hyperparameter choices, reconstruction error distribution, and detection thresholds, along with a discussion of false positives and false negatives.

3.1 Model Training Performance

The Autoencoder was trained exclusively on normal traffic to learn patterns representing benign network behavior. Training was conducted over 50 epochs with a batch size of 32. The training loss (MSE) steadily decreased during the first 30 epochs and stabilized afterward, indicating the model's convergence. Below is the training and validation loss plot, which demonstrates that the model generalizes well without overfitting(Fig.2):

```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.show()
```

Fig. 2. Training and validation loss plot.

- **Training Loss**: 0.0021
- **Validation Loss**: 0.0023

The minor difference between training and validation loss suggests that the model is well-fitted and can generalize to unseen data.

3.2 Reconstruction Error Analysis

Once trained, the Autoencoder was evaluated by feeding the test set (which contains both normal and attack traffic) through the network[8]. The reconstruction error was computed for each sample using Mean Squared Error (MSE). A histogram of the reconstruction errors is shown below to illustrate the distribution of errors for normal and anomalous samples (Fig.3).

```
plt.hist(reconstruction_errors[labels == 'BENIGN'], bins=50, alpha=0.6, label='Normal Traffi
plt.hist(reconstruction_errors[labels != 'BENIGN'], bins=50, alpha=0.6, label='Attack Traffi
plt.xlabel('Reconstruction Error')
plt.ylabel('Number of Samples')
plt.legend()
plt.show()
```

Fig. 3. The distribution of errors for normal and anomalous samples

The plot demonstrates that normal traffic samples generally have lower reconstruction errors, while anomalous samples show higher errors. This separation justifies the use of a threshold to classify samples as either normal or anomalous.

3.3 Threshold Selection and Anomaly Detection

The anomaly detection threshold was calculated based on the mean and standard deviation of reconstruction errors for normal samples(2):

$$Threshold = \mu + 2\sigma \tag{2}$$

This formula balances the sensitivity of the model by capturing most normal samples within the threshold while flagging outliers as anomalies.

Calculated Threshold: 0.0057

Below is the ROC curve, demonstrating the model's performance across varying thresholds(Fig.4):

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, _ = roc_curve(labels, reconstruction_errors)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'ROC curve (area = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

Fig. 4. The model's performance across varying thresholds

3.4 Threshold Selection and Anomaly Detection

The effectiveness of the model was evaluated using the following metrics:

- Accuracy: Proportion of correctly classified samples (both normal and attack traffic).
- Precision: Ratio of true positives to all predicted positives.
- Recall: Ratio of true positives to all actual positives.
- F1-Score: Harmonic mean of precision and recall, balancing these two metrics.
- ROC-AUC: Area Under the Receiver Operating Characteristic curve, which evaluates the model's ability to distinguish between normal and attack traffic.

The following table summarizes the results in Table 1:

Table 1. The effectiveness of the model

Metric	Score
Accuracy	94.5%
Precision	92.3%
Recall	90.1%
F1-Score	91.2%
ROC-AUC	0.95

3.5 Interpretation of Results

High Accuracy and Precision: The model achieved an accuracy of 94.5% and a precision of 92.3%, indicating its ability to accurately classify a large portion of traffic as normal or anomalous[9]. The high precision also suggests that the model effectively minimizes false positives, reducing the number of benign traffic flows flagged as attacks.

Recall and Detection of Subtle Attacks: A recall of 90.1% indicates that the model successfully identifies most attack traffic. However, the model struggled with low-intensity, stealthy attacks, such as slow brute-force attempts. These attacks generated lower reconstruction errors, which were sometimes misclassified as normal traffic, leading to false negatives.

ROC-AUC: The ROC-AUC value of 0.95 suggests that the model performs well across different threshold values, demonstrating strong discriminative ability between normal and anomalous traffic.

3.6 False Positives and False Negatives

The Autoencoder's performance was impacted by the balance between false positives (benign samples classified as attacks) and false negatives (attacks classified as benign). Adjusting the threshold directly affected this trade-off:

- Lower Threshold: Increased detection of attacks but also resulted in more false positives.
- Higher Threshold: Reduced false positives but missed some stealthy attacks, increasing false negatives.

To further mitigate this issue, adaptive thresholding mechanisms can be explored in future work, where thresholds adjust dynamically based on network conditions.

3.7 Comparison with Traditional Methods

While traditional methods such as signature-based IDS rely on predefined rules to detect known attacks, the Autoencoder can identify both known and previously unseen anomalies without explicit labels. This makes it more adaptable to evolving threats, such as zero-day attacks. However, Autoencoders are not without limitations. They require careful tuning of thresholds and hyperparameters to avoid generating too many false positives or missing sophisticated attacks.

The Autoencoder model was compared with other approaches such as Random Forest, SVM, and KNN. Below is a summary of the comparative analysis in Table 2:

Table 2. Comparison of Autoencoder and ML Models

Model	F1-Score	ROC-AUC
Autoencoder	91.2%	0.95
Random Forest	89.7%	0.93
SVM	85.4%	0.90
KNN	82.1%	0.88

The Autoencoder demonstrated superior performance in detecting previously unseen anomalies and adapting to dynamic attack patterns compared to traditional models.

3.8 Summary of Results

The Autoencoder demonstrated high performance in detecting anomalies, achieving a balanced F1-score of 91.2% and a ROC-AUC of 0.95. However, the model's sensitivity to subtle attacks highlights the need for further improvements. The following conclusions can be drawn from the results:

1. **Strengths**: The model performed well in identifying anomalies, particularly large-scale attacks like DDoS and port scans.
2. **Limitations**: Detection of stealthy attacks remains challenging, requiring further tuning or ensemble approaches.
3. **Potential Improvements**: Adaptive thresholding and combining Autoencoders with other models, such as Random Forest or LSTM, could enhance detection capabilities.

4. Conclusions

The discussion section evaluates the performance of the Autoencoder-based anomaly detection model, focusing on its strengths, limitations, practical applications, and recommendations for future improvements. The model's ability to generalize across diverse attack types and its limitations in handling certain scenarios are analyzed. Additionally, potential strategies for overcoming identified challenges are discussed.

4.1 Strengths of the Autoencoder Model

The experimental results demonstrate that the Autoencoder effectively identifies anomalies in network traffic, achieving high precision, recall, and an F1-score of 91.2%. The model's strengths include:

1. **Ability to Detect Both Known and Unknown Threats**: Since the Autoencoder was trained exclusively on normal traffic, it is capable of detecting previously unseen attack types (e.g., **zero-day attacks**) by identifying deviations from expected behavior.
2. **Unsupervised Learning Approach**: The Autoencoder does not rely on labeled attack data, which makes it adaptable to **dynamic environments** where new types of cyberattacks may arise. This is a key advantage over traditional signature-based systems, which are limited to known threats.
3. **High ROC-AUC Score**: The model's **ROC-AUC of 0.95** demonstrates that it maintains high discriminatory power across various threshold settings, making it suitable for deployment in production environments with adjustable sensitivity.
4. **Minimal Feature Engineering**: The Autoencoder requires little feature selection or manual tuning of inputs, which reduces the complexity of building and maintaining the model over time.

4.2 Limitations and Challenges

Despite its strong performance, the Autoencoder model exhibited some limitations:

1. **Sensitivity to Threshold Settings**: A fixed threshold, though effective during evaluation, may not work optimally in real-time network environments where traffic patterns fluctuate. Setting the threshold too low increases false positives, while a high threshold may lead to missed attacks.
2. **False Negatives with Subtle Attacks**: The model struggled to detect low-intensity or stealthy attacks such as slow brute-force attempts, which generated reconstruction errors close to normal traffic. These attacks are designed to evade detection by generating minimal deviations from expected behavior.
3. **Imbalanced Attack Distribution**: Some attack types in the CICIDS2017 dataset (e.g., DDoS) were more prevalent than others (e.g., infiltration attacks), which may have impacted the model's ability to generalize equally well across all attack types.
4. **Lack of Adversarial Robustness**: Like most machine learning models, the Autoencoder is vulnerable to adversarial attacks, where an attacker manipulates input data to deceive the model. This presents a significant challenge in cybersecurity applications.

4.3 Practical Implications

The ability to detect anomalies in real-time makes Autoencoder models highly relevant for modern intrusion detection systems (IDS). These systems need to handle large volumes of network traffic while identifying both known and unknown threats without human intervention. The Autoencoder's low dependency on labeled data also reduces the effort required for data annotation and maintenance over time.

In practical deployments, the model could serve as the first layer of defense in a network security framework, flagging suspicious activities for further inspection by signature-based systems or security analysts. Ensemble models, which combine the Autoencoder with other techniques like Random Forest or LSTM, could further enhance detection accuracy and reduce false positives.

4.4 Recommendations for Future Improvements

Several strategies could be adopted to enhance the model's robustness and adaptability:

1. **Adaptive Thresholding**: Implementing adaptive threshold mechanisms that adjust the threshold dynamically based on network traffic patterns could reduce false positives and improve the model's effectiveness in real-time environments.
2. **Hybrid and Ensemble Models**: Combining the Autoencoder with other machine learning models, such as Random Forest or Long Short-Term Memory (LSTM) networks, could improve detection accuracy, especially for time-dependent attack patterns.
3. **Handling Data Imbalance**: Techniques such as oversampling underrepresented attack types or using synthetic data generation can improve the model's performance across all categories of attacks.
4. **Adversarial Defense Mechanisms**: Future work should focus on developing adversarial training methods or robust architectures to mitigate the impact of adversarial attacks on the Autoencoder model.
5. **Feature Augmentation**: Additional network features, such as payload content or encrypted traffic patterns, could be integrated into the model to enhance its ability to detect more sophisticated threats.

4.5 Summary

This study demonstrates the potential of Autoencoder neural networks as a versatile and efficient anomaly detection tool for modern network environments. Key strengths of the model include its capability to detect both known and previously unseen threats, reliance on minimal labeled data, and straightforward integration into existing cybersecurity systems. Despite these advantages, certain limitations were identified, such as sensitivity to threshold settings, false negatives for stealthy attacks, and susceptibility to adversarial manipulation.

To address these challenges, future work should explore the incorporation of adaptive thresholds tailored to network dynamics, hybrid models that combine Autoencoders with supervised learning techniques, and adversarial defense mechanisms to mitigate vulnerabilities. Additionally, addressing data imbalance through advanced sampling strategies or the integration of synthetic data could improve detection performance for rare or stealthy attacks.

Looking ahead, Autoencoders hold significant promise as a foundational component of AI-driven cybersecurity frameworks. Their integration with broader systems, such as Security Information and Event Management (SIEM) platforms, can facilitate proactive and scalable threat detection across diverse network environments. Furthermore, collaborative research into ensemble methods and real-time deployment strategies will be critical for ensuring the long-term reliability and resilience of these models in combating the evolving landscape of cyber threats.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. DOI: 10.5555/3086952.

[2] M. Cicirello, and M. Pontani. “Deep Learning in Cybersecurity: A Review.” *Journal of Network Security*, vol. 32, no. 4, pp. 124–135, 2019. DOI: 10.1016/j.jns.2019.07.006.

[3] Canadian Institute for Cybersecurity. “CICIDS2017 Dataset.” University of New Brunswick. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed Oct. 10, 2024).

[4] M. Abou-Elhamayed, S. J. Hussain, F. Al-Turjman, and O. K. Ewees. “Anomaly Detection in Network Traffic Using Autoencoders.” *IEEE Transactions on Cybersecurity*, vol. 17, no. 2, pp. 67–78, 2021. DOI: 10.1109/TCS.2021.3053349.

[5] K. Ahmed, A. Shahid, F. Usman, and M. Zahid. “AI-Driven Cybersecurity Frameworks: An Overview.” *ACM Computing Surveys*, vol. 55, no. 3, pp. 1–29, 2023. DOI: 10.1145/3543879.

[6] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami. “Practical Black-Box Attacks Against Machine Learning.” *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIACCS)*, pp. 506–519, 2017. DOI: 10.1145/3052973.3053009.

[7] A. Roy, S. Sharma, A. Bose, and A. Mukherjee. “Towards Robust Anomaly Detection: Adversarial Training for Autoencoders.” *Information Sciences*, vol. 602, pp. 150–170, 2022. DOI: 10.1016/j.ins.2022.04.016.

[8] I. Zavushchak, Z. Rybchak. “AI-Powered Tools for Data Privacy Enhancement in Cybersecurity Systems.” *Cybersecurity and Privacy Journal*, vol. 11, no. 1, pp. 35–47, 2022. DOI: 10.1016/j.cybsec.2022.01.004.

[9] I. Zavushchak, Z. Rybchak, and P. Dovganych. “Anomaly Detection in IoT Networks Using Autoencoders: A Comparative Study.” *International Journal of Computer Science and Network Security*, vol. 20, no. 12, pp. 45–55, 2023. DOI: 10.22937/IJCSNS.2023.012.

Authors’ Profiles

Iryna Zavushchak obtained a master's degree in the specialty "Artificial Intelligence Systems" and obtained the qualification "Computer Systems Analyst" of the Lviv Polytechnic National University in 2015. Obtained the degree of candidate of technical sciences, majoring in mathematics and software support of computing machines and systems, from the National University of Lviv Polytechnic in 2019. Currently works as an associate professor at the Department of Information Systems and Networks of the Lviv Polytechnic National University.

How to cite this paper: Iryna Zavushchak, "The Impact of Artificial Intelligence on Cybersecurity and Data Protection", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.15, No.4, pp. 65-72, 2025. DOI:10.5815/ijwmt.2025.04.05