Autoencoder-Based Anomaly Detection in Networks

Yavuzalp Kaplan, Johannes Späth*, Max Helm*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: yavuzalp.kaplan@tum.de, spaethj@net.in.tum.de, helm@net.in.tum.de

***Abstract*—Modern computer networks produce large amounts of traffic, and unusual patterns in this traffic can signal failures, misconfigurations, or security threats. Autoencoders are used for anomaly detection, as they capture normal patterns and identify anomalies through reconstruction error. In this paper, we review the use of autoencoder-based models for network anomaly detection. We discuss preprocessing techniques, loss functions, and hyperparameters, and we summarize experimental results from previous studies on datasets such as NSL-KDD, UNSW-NB15, KDDCUP'99, and InSDN. The results show that while autoencoders achieve promising detection performance, outcomes vary depending on the dataset and parameter choices.**

***Index Terms*—Autoencoder, Anomaly Detection, Preprocessing, Activation Functions, Loss Functions**

1. Introduction

In October 2016, a large Distributed Denial of Service (DDoS) attack caused major problems across the internet [1]. Websites like Twitter, Netflix, Reddit and many others became unreachable for several hours. The attack was executed using a botnet called Mirai, which had infected many Internet of Things (IoT) devices such as digital cameras and DVR players. These devices, which normally send very little traffic, began to generate large amounts of repeated Domain Name System (DNS) requests to Dyn, a major DNS provider. Traditional intrusion detection systems (IDS), which usually detect known attack patterns, failed to notice the early signs of this attack because the traffic did not match any known signatures. But this kind of abnormal network behavior, especially many quiet devices sending a lot of similar traffic at the same time, could have been detected earlier using machine learning methods like autoencoders.

Attacks like the Mirai botnet show that network security is becoming more important. Today, cyberattacks happen more often and are harder to detect [2]. Attacks such as DoS, port scans, data theft can cause serious damage, e. g., by disrupting critical services, mapping network vulnerabilities, or exfiltrating sensitive information. Traditional IDS methods use signatures to compare traffic to known attack types. This helps with known threats, but it does not work well for new or unknown attacks, which are called zero-day attacks [3].

Anomaly detection offers a different approach. Instead of relying on predefined attack signatures, it tries to find unusual behavior in the network. This makes it useful for detecting new and unknown threats.

In this paper, we explain how autoencoders can be used for network anomaly detection. We describe the preprocessing steps, how the autoencoder model works and how reconstruction loss helps to detect anomalies. We also discuss the types of attacks that autoencoder-based models can find and look at results from previous studies.

2. Background

Autoencoders are a type of neural network that are used for anomaly detection. They work by learning normal patterns in the data. An autoencoder has three parts: an encoder, which compresses the input into a smaller representation, a latent space that stores this compressed form and a decoder, which reconstructs the original data from it. The model is trained to minimize the difference between the input and the output. If the model sees something very different from what it has learned, the difference, or reconstruction loss, becomes larger, which signals a possible anomaly.

However, network data often contains different types of features. Some are strings, like `protocol_type` and `flag`, while others are numbers, like `src_bytes` and `dst_bytes`. Some are binary, like `is_guest_login`. These different types of data cannot be used directly in an autoencoder. Preprocessing steps are needed to convert them into numbers. One common method is *One-Hot-Encoding*, which turns categorical features into binary vectors. Datasets like KDDCUP'99 and NSL-KDD provide examples of such features and are often used in network anomaly detection research [4].

Several previous studies [3], [5]–[14] have applied autoencoder-based models to network anomaly detection tasks, using public datasets such as NSL-KDD, KDDCUP'99, UNSW-NB15, InSDN, etc. These works show that autoencoders can successfully detect a variety of attacks, including DoS, Probe, R2L, U2R, Backdoors, Exploits, Fuzzers and more. However, the performance of these models depends on many factors, such as the choice of hyperparameters, the preprocessing techniques and the characteristics of the dataset [15].

3. Autoencoders

An autoencoder is a type of machine learning model based on a neural network architecture. Its main goal is to reconstruct given input. It is a structure that learns to

 doi: 10.2313/NET-2025-11-3_17

compress the input data into a lower-dimensional space, and then learns to make as faithful an approximation of the input data as possible from this compressed representation. In the process, the model learns the key features and patterns of the data, filtering out unnecessary information and noise.

The structure of an autoencoder usually consists of three main parts: encoder, latent space and decoder. The encoder part takes the input data such as an image, an audio file or a network connection log and transforms it into a smaller and denser representation, the latent space, also known as bottleneck. This latent space is a compressed and informative representation of the input data. The decoder takes the compressed representation and tries again to produce a prediction of the original input data. Figure 1 illustrates the simple architecture of an autoencoder, including the encoder, latent space and decoder components [16].

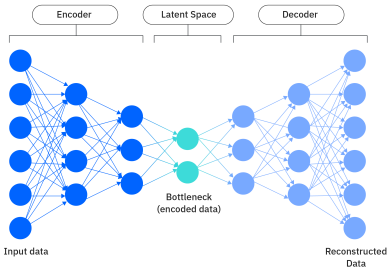

Figure 1: Simple Autoencoder Architecture

3.1. Learning Techniques

Within the concept of machine learning, there are different learning techniques. These are divided into supervised, semi-supervised and unsupervised learning. Supervised learning is a type of learning where each input data has a corresponding label. The model learns the relationship between the input data and the label, and aims to predict the label for new data. Semi-supervised learning is an approach that combines labeled and unlabeled data to improve learning accuracy when labeled data is scarce.

However, these methods fail to detect zero-day attacks that can occur daily in networks [3]. For this reason, an unsupervised learning technique is applied to the autoencoder for anomaly detection in networks. In this technique, the input data is not labeled, but is learned from the data itself. The model tries to discover patterns, relationships, groupings or hidden structures in the data. Since there are no labels, the model learns entirely from the distribution of the data and tries to discover normal and abnormal states or clusters within the data.

3.2. Activation Functions, Encoder and Decoder

Activation functions enable the network to build a nonlinear structure and help it learn complex data patterns. Which activation function is used for encoding and decoding the data directly affects the success and the performance of the model. Therefore, this function, which determines what and how the model learns, should be carefully selected before training [15]. If the activation function is not chosen correctly, the model may struggle to learn complex relationships and may not produce the expected results. Commonly used activation functions are sigmoid, tanh, rectified linear unit (ReLU), and scaled exponential linear unit (SELU), which are defined in Eq. (1) – (4).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3}$$

$$\mathrm{SELU}(x) \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^{x}-1) & \text{if } x \leq 0 \end{cases} \tag{4}$$

We will now focus on how the encoder and decoder work with activation functions. An encoder is a neural network that compresses the input data into a lower-dimensional latent space, i. e., it maps a high-dimensional input vector X into a smaller latent vector Z.

$$Z = \sigma(WX + b) \tag{5}$$

Here, Z is the compressed representation in latent space, σ is the activation function, W is the weight matrix and b is the bias vector. From this latent representation produced by the encoder, the decoder tries to produce a prediction of the input data. The goal is to output $\hat{X}$ as close as possible to the input X [5].

$$\hat{X} = \sigma(W'Z + b') \tag{6}$$

4. Anomaly Detection

As discussed in Section 3, the main purpose of autoencoders is to compress the input data into a smaller latent space and then reconstruct the original data from this representation. In this process, the success of the model is measured by the similarity between the input data and the reconstructed data. One of the most common methods used to measure this similarity is reconstruction loss [15].

4.1. Loss Functions

Reconstruction loss is a numerical measure of how accurately the model can reproduce a data given as input. Functions such as Mean Squared Error (MSE) or Binary Cross-Entropy (BCE) are usually preferred [15]. MSE is a common choice, especially when working with numerical data. MSE takes the average of the squares of the differences between the input and model output and is defined as follows:

$$L_{\mathrm{MSE}}(X, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2 \tag{7}$$

Where n is the number of features, X_i is the i-th feature in the original input and $\hat{X}_i$ is the i-th feature in the

 doi: 10.2313/NET-2025-11-3_17

reconstruction output. Alternatively the BCE loss function can also be applied. This function, which is used when the input data is binary (0 or 1), calculates the binary cross-entropy loss difference between the expected and actual output.

$$L_{\text{BCE}}(X, \hat{X}) = -\frac{1}{n}\sum_{i=1}^{n}\Big(X_i \cdot \log(\hat{X}_i) + (1 - X_i) \cdot \log(1 - \hat{X}_i)\Big) \quad (8)$$

The loss function should be chosen carefully so that the model can learn correctly. A loss function that is not appropriate for the data type or the problem can negatively affect the model's learning process and cause inaccurate results. For example, using BCE loss in numerical data can lead to the model's outputs being stuck in a wrong range. Hence, the loss function must match the data type and the model's purpose.

4.2. Preprocessing

Network logs often contain various types of features, such as `protocol_type`, `src_bytes`, `dst_bytes`, `flag`, `is_guest_login`, `num_access_files` etc. Public datasets like KDDCUP'99, NSL-KDD, UNSW-NB15 include such examples [4]. However, these raw data features cannot be directly fed into an autoencoder model, because some of them are strings like `protocol_type`, some are numeric like `src_bytes`, some are binary like `is_guest_login`. Autoencoders only use numerical data for training and testing [17]. To make them suitable for training, a preprocessing step is necessary. One common approach is to use *One-Hot-Encoding*, which converts categorical and string-based features into numerical vectors. This process transforms all data into a fully numerical format, allowing the autoencoder to process the input effectively.

For example, the feature `protocol_type` has three distinct attributes: `tcp`, `udp` and `icmp`, each of which is encoded into a three-dimensional binary vector, such as [1,0,0], [0,1,0] and [0,0,1], respectively. In other words, the single feature `protocol_type` is encoded into three features by one-hot-encoding [6].

After preprocessing, the original data is transformed into a fully numerical vector. A simplified example of such a vector is shown in (9).

$$x = [0.0,\ 1.0,\ 0.78,\ 0.54,\ 0.12,\ 0.0,\ 1.0,\ 0.33,\ \ldots] \quad (9)$$

4.3. Security

Anomaly detection models are important tools for protecting computer networks. They find unusual activities that are different from normal network behaviour. Autoencoder-based models are especially useful because they learn what normal traffic looks like and detect anything that seems unusual, even if the attack is new or unknown. Unlike signature-based systems, they do not need labeled attack data and can work without knowing the exact type of attack in advance.

These models detect many different kinds of network attacks by checking how much a new traffic pattern differs from what was learned during training. The most common attack types that anomaly detection systems aim to protect against include Denial of Service (DoS), Probe, Remote to Local (R2L), User to Root (U2R), Backdoors, Exploits, Fuzzers, Generic, Portscans, Reconnaissance, Shellcode, and Worms [18].

Autoencoder-based models detect these attacks by spotting patterns that do not match normal traffic, helping to protect networks from a variety of threats. The NSL-KDD and KDDCUP'99 datasets are commonly used in tests for detecting DoS, Probe, R2L and U2R attacks, while the UNSW-NB15 dataset is used to evaluate models against attacks such as Backdoors, DoS, Exploits, Fuzzers, Generic, Portscans, Reconnaissance, Shellcode and Worms [18].

5. Evaluation

Evaluating the performance of an anomaly detection system is essential to understand how efficiently it can detect unusual patterns in network traffic. This section introduces the commonly used performance metrics in anomaly detection and discusses the factors that affect model performance, such as the choice of loss function, network architecture, and training settings.

5.1. Evaluation Metrics

To evaluate the performance of autoencoder-based anomaly detection models, four common metrics are used: *precison*, *recall*, *f1-score*, and *accuracy*. These metrics are computed from four basic measures, true positive (TP), false positive (FP), true negative (TN), false negative (FN):

- TP, defined as the number of anomalies correctly identified as anomalies.
- FP, referring to the number of normal samples incorrectly identified as anomalies.
- TN, defined as the number of normal samples correctly identified as normal.
- FN, referring to the number of anomalies incorrectly identified as normal.

The metrics for performance measurement in anomaly detection systems are as follows:

Precision: Indicates the proportion of samples predicted as anomalies that are actually anomalies.

$$\textit{Precision} = \frac{TP}{TP + FP} \quad (10)$$

Recall (Sensitivity): Measures the proportion of actual anomalies correctly identified by the model.

$$\textit{Recall} = \frac{TP}{TP + FN} \quad (11)$$

F1-Score: The harmonic mean of *Precision* and *Recall*.

$$F_1 = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}} \quad (12)$$

Accuracy: The overall correct prediction rate of the model.

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

 doi: 10.2313/NET-2025-11-3_17

5.2. Hyperparameters

The performance of autoencoder-based anomaly detection systems depends on various factors and hyperparameters [15], including:

- **Number of hidden layers:** More layers can capture complex patterns but may also lead to overfitting, where the model memorizes the training data and fails to generalize.
- **Number of neurons per layer:** Controls the model's ability to learn patterns; more neurons improve learning but increase computational cost.
- **Size of the latent space:** A smaller space forces stronger compression and may help anomaly detection, but too small a space can cause information loss.
- **Activation function:** The choice of activation functions affects the model's ability to capture nonlinear relationships.
- **Loss function:** Defines the optimization target; MSE and BCE are the most common choices.
- **Learning rate:** Sets the update step size; high rates can cause instability, while low rates may slow convergence.
- **Number of epochs:** The number of training cycles; too few may lead underfitting, where the model fails to capture patterns in the data, while too many may result in overfitting.
- **Batch size:** Number of samples used per training step; large batches provide more stable updates, while small batches add variability but may improve generalization.

These hyperparameters must be carefully tuned to balance model complexity, training efficiency and detection performance.

5.3. Experimental Results

Anomaly detection tasks have been extensively studied in the literature, and several works have demonstrated the potential of autoencoder-based models in identifying anomalous network traffic. This section provides a summary of experimental results reported in previous studies, highlighting the performance of autoencoder architectures across different datasets and metrics.

Table 1 summarizes the results on the NSL-KDD dataset. Study [3] tested a memory-augmented deep autoencoder (MemAE) to improve anomaly detection by using a memory mechanism. Study [7] applied a sparse autoencoder to learn sparse features for better detection. Study [6] chose the Mean Absolute Error (MAE) as loss function for better performance, while Study [8] tested a denoising autoencoder (DAE) to make the model more robust to noise. Additionally, Study [9] applied a Fully Connected Network (FCN) and a Shallow Long Short-Term Memory network (SLSTM), achieving strong results in terms of precision and recall.

Table 2 covers the UNSW-NB15 dataset. Study [3] tested MemAE again on this dataset. Additionally, Study [10] used an autoencoder for different attacking scenarios such as DoS, Backdoors, Shellcode, etc. Study [11] evaluated other machine learning models on this dataset, including Decision Tree (DT), K-Nearest Neighbours (KNN), Naïve Bayes (NB), and Random Forest (RF). Furthermore, Study [12] explored the use of more complex models such as Stacked Sparse Autoencoder (SSAE), Variational LSTM (VLSTM), and CNN-LSTM, highlighting the diversity of approaches in handling anomaly detection tasks.

Table 3 presents results for the KDDCUP dataset. For example, Study [13] used an autoencoder model on this dataset. Study [14] used the Deep Autoencoding Gaussian Mixture Model (DAGMM) on the same dataset, combining clustering and autoencoder reconstruction errors to detect anomalies.

Table 4 shows results on the InSDN dataset. Study [5] applied both a One-Class Support Vector Machine (OC-SVM) and a Long Short Term Memory autoencoder (LSTM) on this dataset.

TABLE 1: Evaluation Metrics on NSL-KDD Dataset

Model	Accuracy	Precision	Recall	F1-Score
MemAE [3]	0.8951	0.9062	0.8951	0.8993
Sparse AE [7]	0.8839	0.8544	0.9595	0.904
AE [6]	**0.9061**	0.8683	0.9843	0.9226
AE [8]	0.8828	0.9123	0.8786	0.8951
DAE [8]	0.8865	0.9648	0.8308	0.8928
FCN [9]	-	**0.997**	0.874	0.931
SLSTM [9]	-	0.983	**0.996**	**0.99**

TABLE 2: Evaluation Metrics on UNSW-NB15 Dataset

Model	Accuracy	Precision	Recall	F1-Score
MemAE [3]	0.853	**0.8774**	0.853	0.8526
AE [10]	**0.9516**	-	-	-
DT [11]	0.8455	0.864	0.846	0.8549
KNN [11]	0.8449	0.855	0.845	0.85
NB [11]	0.7639	0.782	0.764	0.7729
RF [11]	0.8363	0.869	0.836	0.8522
SSAE [12]	-	0.731	0.963	0.832
VLSTM [12]	-	0.86	**0.978**	**0.907**
CNN-LSTM [12]	-	0.801	0.956	0.872

TABLE 3: Evaluation Metrics on KDDCUP Dataset

Model	Accuracy	Precision	Recall	F1-Score
AE [13]	0.9282	0.9236	**0.9923**	**0.96**
DAGMM [14]	-	**0.9297**	0.9442	0.9369

TABLE 4: Evaluation Metrics on InSDN Dataset

Model	Accuracy	Precision	Recall	F1-Score
OC-SVM [5]	0.875	0.89	**0.93**	0.91
LSTM [5]	**0.905**	**0.93**	**0.93**	**0.93**

6. Conclusion and Future Work

Autoencoder-based models are a useful tool for detecting anomalies in network traffic by learning normal patterns and identifying unusual data points. These models can help identify various types of attacks, including both

 doi: 10.2313/NET-2025-11-3_17

known and unknown threats. However, the performance of an autoencoder model depends on careful preprocessing, the choice of loss function, and tuning of hyperparameters.

The experimental results summarized in Tables 1 – 4 show that no single autoencoder-based model consistently outperforms others across all datasets and metrics. For example, on the NSL-KDD dataset, FCN achieved the best precision, while SLSTM reached the highest recall. On UNSW-NB15, MemAE achieved the highest precision, but VLSTM provided the strongest recall. For KDDCUP, AE and DAGMM both delivered strong results, with AE performing better in recall and DAGMM achieving the highest precision. On the InSDN dataset, LSTM clearly outperformed OC-SVM across all metrics. These findings suggest that the dataset and selected hyperparameters play a major role in determining which model performs best.

While existing studies show promising results, further research is needed to optimize models for real world network environments and large scale data. Future work should focus on making these models more scalable, more robust, and easier to deploy in practice.

References

[1] N. Woolf, "DDoS Attack That Disrupted Internet Was Largest of Its Kind in History, Experts Say," https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, 2016, [Online; accessed 18-Jun-2025].

[2] J. Martin, "Cybersecurity Statistics: IoT, DDoS, and Other Attacks," https://explodingtopics.com/blog/cybersecurity-stats#iot-ddos-and-other-attacks, 2025, [Online; accessed 19-Aug-2025].

[3] B. Min, Y. Jihoon, S. Kim, D. Shin, and D. Shin, "Network Anomaly Detection Using Memory-Augmented Deep Autoencoder," *IEEE Access*, vol. PP, pp. 1–1, 07 2021.

[4] M. S. Yadav and R. Kalpana, "Data Preprocessing for Intrusion Detection System Using Encoding and Normalization Approaches," in *2019 11th International Conference on Advanced Computing (ICoAC)*, 2019, pp. 265–269.

[5] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network Anomaly Detection Using LSTM Based Autoencoder," in *Proc. 16th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, Alicante, Spain, 2020, pp. 37–45.

[6] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset," *IEEE Access*, vol. PP, pp. 1–1, 2021.

[7] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," *EAI Endorsed Transactions on Security and Safety*, vol. 3, no. 9, 2016.

[8] R. C. Aygun and A. G. Yavuz, "Network Anomaly Detection with Stochastically Improved Autoencoder Based Models," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 193–198.

[9] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An Empirical Evaluation of Deep Learning for Network Anomaly Detection," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, 2018, pp. 893–898.

[10] L. Ashiku and C. Dagli, "Network Intrusion Detection System using Deep Learning," *Procedia Computer Science*, vol. 185, pp. 239–247, 2021, big Data, IoT, and AI for a Smarter Future.

[11] F. A. Khan and A. Gumaei, "A Comparative Study of Machine Learning Classifiers for Network Intrusion Detection," in *Artificial Intelligence and Security*, X. Sun, Z. Pan, and E. Bertino, Eds. Cham: Springer International Publishing, 2019, pp. 75–86.

[12] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM Enhanced Anomaly Detection for Industrial Big Data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2021.

[13] M. Ahmed, A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[14] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection," in *International Conference on Learning Representations*, 2018.

[15] K. Berahmand, F. Daneshfar, E. S. Salehi, and et al., "Autoencoders and their applications in machine learning: a survey," *Artificial Intelligence Review*, vol. 57, no. 28, 2024.

[16] IBM, "Variational Autoencoder," https://www.ibm.com/de-de/think/topics/variational-autoencoder, 2024, [Online; accessed 26-May-2025].

[17] H. Torabi, S. Mirtaheri, and S. Greco, "Practical Autoencoder-Based Anomaly Detection by Using Vector Reconstruction Error," *Cybersecurity*, vol. 6, 2023.

[18] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.