Science World Journal Vol. 20(No 4) 2025 https://dx.doi.org/10.4314/swj.v20i4.8
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

Full Length Research Article

AUTOENCODER-BASED MODEL FOR DETECTING IOT NETWORK TRAFFIC ANOMALIES

*[1]Bako Halilu Egga, [1]Abdullahi Salihu Audu, [1]Gilbert O.I Aimufua, [1]Morufu Olalere, [1]Binyamin Adeniyi Ajayi, [2]Igila Terngu Solomon

[1]Department of Computer Science, Nasarawa State University, Keffi, Nasarawa State, Nigeria
[2]Infoquest Academy, Suite A-305 AGY Plaza, Bakan Gizo, Keffi, Nasarawa State, Nigeria

*Corresponding Author Email Address: bakohaliluegga@nsuk.edu.ng

ABSTRACT
Cyberattacks on computer networks are becoming increasingly sophisticated, particularly in Internet of Things (IoT) environments, where devices continuously generate large and complex amounts of data. Traditional security systems that rely on predefined rules or signatures often fail to detect new or evolving threats. Even deep learning methods, such as RNNs and CNNs, face challenges in handling dynamic traffic efficiently. To address these issues, this study introduces an autoencoder-based anomaly detection model that learns to identify abnormal network activities. The model was trained using Kaggle datasets containing both normal IoT traffic and malicious traffic from well-known botnets like Mirai and BASHLITE. By compressing network data into a latent space and reconstructing it, the model uses reconstruction error to detect unusual patterns that indicate anomalies. The experimental results were highly promising, achieving 99% accuracy, precision, recall, and F1-scores above 99%**.** Unlike previous studies that depend on simulated or cloud-based data, this research highlights the power of autoencoders for real-world IoT anomaly detection and lays a strong foundation for developing real-time intrusion detection systems.

Keywords: Autoencoder, Anomaly Detection, Internet of Things (IoT), Intrusion Detection System (IDS), Machine Learning.

INTRODUCTION

The widespread availability of Internet of Things (IoT) devices has revolutionized contemporary life, making it easier to connect everything in smart homes, healthcare, transportation, and industrial networks (Qiu *et al.,* 2025). Yet this rapid penetration has created serious cybersecurity issues. One of the emergent threats is the susceptibility of IoT devices to botnet attacks, which take advantage of resource-limited architectures and poor security settings. Malware types such as Mirai and BASHLITE were responsible for mass-scale distributed denial-of-service (DDoS) attacks that threatened to compromise the stability and reliability of critical infrastructures (Qiu *et al.,* 2025).

Despite the scope of the problem, the academic community is marred by an after-all shortage- the lack of realistic and publicly available botnet datasets that can potentially aid in intrusion and anomaly detection model development and evaluation. Those datasets that are available tend to be antiquated, artificial, or fail to adequately represent real-world IoT traffic, thus weakening and limiting the scope of put-forward security interventions (Ayad *et al.,* 2024). This limitation has weakened the creation of scalable and reliable anomaly detection processes capable of mitigating emerging IoT attacks.

Recent works have sought to overcome this challenge by using deep learning and hybrid detection methods. Somma (2025) is a case in point, with the study exploring temporal differential consistency autoencoders for enhanced anomaly detection in cyber-physical systems. The same goes for Shah et al. (2024), who proposed a hybrid deep learning model for IoT botnet detection. All these efforts indicate the urgency of the issue, coupled with the new solutions being explored. However, there are concerns with dataset variability, real-time tunability, and adaptability to evolving attack patterns (Shah *et al.,* 2024).

This study is motivated by the immediate necessity of bridging the dataset gap in IoT botnet research. With real traffic records of compromised commercial IoT devices infected with Mirai and BASHLITE, it would provide a more realistic platform for the evaluation of anomaly detection models. In toto, the paper fulfills its contribution towards the advancement of IoT security through bridging the gap between theoretical model development and practical applicability.

Present studies have shown major advancements in anomaly detection systems for IoT and networks with attempts for higher precision, computational complexity, and feature abstraction. Alaghbari et al. (2023) suggested a deep autoencoder-based combined model that focused on detecting anomalies and feature abstraction together. Their approach demonstrated strong detection performance with reduced computational complexity compared to OC-SVM (One-Class Support Vector Machine) and Isolation Forest algorithms. However, their model was primarily static network information-based and failed to fully explore real-time flexibility or temporal modeling of behavior.

Rhachi, Balboul, and Bouayad (2023) emphasized the growing security concerns in Internet of Things (IoT) networks, more particularly, how challenging it is to determine anomalies due to the unpredictability and dynamic nature of these environments. Their paper suggested a deep autoencoder (DAE) with ANOVA F-test feature selection to enhance the precision in the detection of anomalies. While their approach improved detection performance 85% and 92% for binary and multi-class, respectively, the study was based mostly on the NSL-KDD dataset, which is not representative of modern real-world IoT traffic. Moreover, deployment challenges on embedded or limited-resource devices are yet to be thoroughly investigated.

Torabi et al. (2023) explored a smarter way to detect unusual network activity by using an autoencoder model that measures how much the system's reconstruction of data differs from the original. Instead of treating reconstruction error as a single value, they looked at it feature by feature, which made the model better at spotting subtle irregularities in cloud network traffic. Their experiments on the CIDDS-001 dataset showed clear improvements in accuracy, recall, and F1-score compared to older methods.

What makes their work stand out is the way they refined how

reconstruction errors are calculated and used to detect anomalies. However, their model was only tested in controlled cloud environments. This leaves an open question about how well such a system would perform in real-world or IoT settings, where network data is often more complex and unpredictable. This gap suggests a need for future research to adapt and test similar approaches in dynamic, real-time scenarios.

Ali et al. (2024) introduced an innovative hybrid deep learning model that combines Long Short-Term Memory (LSTM) Autoencoders and Multilayer Perceptrons (MLP) to improve botnet detection in IoT settings. The model showed impressive accuracy of 99.77% and 99.67% on the N-BaIoT2018 and UNSW-NB15 datasets, respectively, performing better than conventional detection schemes. This achievement highlights the increasing importance of merging sequential pattern learning (via LSTM) with non-linear feature extraction (via MLP) for identifying intricate attack patterns.

Although the research attained great detection accuracy, it also identified several limitations that limit scalability and real-world applicability. These are server dependence, hybrid model interpretability, and vulnerability to new attacks. Additionally, system centralization also poses potential dangers to data privacy and robustness since server compromise would expose sensitive model parameters. Decentralized and light-weight designs were proposed by the authors as future alternatives for the purposes of increasing resource-limited IoT devices' adaptability.

Following these findings, Qiu et al. (2025) introduced FedAware (Federated Learning-based Intrusion Detection Model for IoT), an IoT intrusion detection system founded on federated learning that integrates a Fractal Shrinking Autoencoder (FSAE) and one-class SVDD classification. The ImbalMSE (Imbalanced Mean Squared Error) algorithm by them improved model aggregation in non-iid (Non-Independent and Non-Identically Distributed) devices to manage heterogeneity and limited resources in distributed environments. Despite high performance, FedAware did not utilize advanced privacy-preserving methods or adversarial attack robustness testing.

As a supplement to this computational method, Shah et al. (2024) employed fractal–fractional calculus in the modeling of nonlinear re-infection dynamics for infections such as COVID-19. The study extended traditional differential equation models by the introduction of fractal and fractional operators for expressing anomalous diffusion processes as well as complex biological interactions. Employing techniques such as the Volterra–Lyapunov method and fixed-point theory, the study derived stability, sensitivity, and numerical confirmation through comparative analysis based on real data. While robust, the research highlighted the need to extend the model to include additional compartments (e.g., quarantined, exposed, vaccinated) and capture higher-order dynamical interactions for more realistic simulations.

Based on those foundations, Somma (2025) introduced a Hybrid Temporal Differential Consistency Autoencoder (hTDC-AE) to identify anomalies in cyber-physical systems. Through the integration of deterministic and statistical nodes into an autoencoder structure, the model utilized physics-driven consistency principles in machine learning. This allowed the system to detect both gradual and sudden anomalies yet remain computationally efficient enough for edge devices. The hTDC-AE reported faster detection speed (3% improvement compared to prior benchmarks) and interpretability, but also requested deeper exploration of the theoretical consistency of static and dynamic latent features.

To these advances, Al-Qudah (2025) offered a multi-step comparative approach that rigorously assessed preprocessing and modeling settings like RNN-LSTM, autoencoders, and Gradient Boosting for anomaly detection in IoT. His findings revealed that Gradient Boosting yielded stable accuracy regardless of configurations, but autoencoders fared best on recall, a critical factor against false negatives. The work contributed a structured experimental foundation for comprehending the interaction between model selection and preprocessing, albeit largely comparative rather than generative, with open issues regarding adaptive model integration and automation.

Similarly, Ayad et al. (2025) introduced an efficient real-time IoT anomaly detection model via integration of a one-class asymmetric stacked autoencoder and deep neural network (DNN). Their hybrid method retained superior performance, 99.99% accuracy and 0.27 s detection time on the BoT-IoT dataset and performed exceedingly well in real-time scenarios. Nevertheless, despite the study mentioning computational efficiency as well as class imbalance, it only accounted for IoT settings and not temporal feature relationships or cross-domain adaptability in video-based or sensor surveillance applications.

Synthesis and Gap

The literature review suggests three significant gaps:

Dataset Limitations: Synthetic or outdated datasets reduce the usability and reliability of detection models.

Model Transparency and Generalizability: The majority of models perform exceptionally well without addressing any potential dataset bias or providing an interpretable thresholding technique.

Practical Deployment: Few techniques consider the computational feasibility of deploying autoencoder-based anomaly detection in practical IoT scenarios.

This work fills these gaps with a publicly available IoT botnet data set that was collected from infected commercial devices with Mirai and BASHLITE, which is realistic for evaluation. It also identifies accurate anomaly thresholding methods and critically evaluates model performance regarding possible dataset biases, thereby creating robust, realistic, and transparent IoT network anomaly detection systems.

MATERIALS AND METHODS

This research focuses on a deep learning technique known as an Autoencoder, implemented using the Python programming language. The study employed secondary data collected from Internet of Things (IoT) devices, sourced from Kaggle (https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset). The dataset contains traffic data from nine commercial IoT devices that were deliberately infected with Mirai and BASHLITE botnets. These devices were used to capture real network traffic patterns, including both benign and malicious activities. The dataset comprises multivariate sequential data with a total of 7,062,606 instances and 115 attributes.

Dataset Analysis

The data used in this study is network traffic data, which contains various forms of attacks and also benign traffic. The "*type*" column labels each record as its respective class.

Data analysis shows a strong skewing of the distribution of traffic

Science World Journal Vol. 20(No 4) 2025 https://dx.doi.org/10.4314/swj.v20i4.8
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

classes as seen in the bar plot and pie chart. The 'mirai_syn' type of attack was the most dominant, with 30,643 instances (16.1%), closely followed by *'mirai_ack'* (25,549; 13.4%), *'mirai_udp'* (23,766; 12.4%), and *'mirai_udpplain'* (22,135; 11.6%). All these four classes, together, make up a large majority of the dataset.

On the other hand, benign traffic is the least frequent category with a paltry 12,387 instances (6.5%), showing an obvious imbalance. The '*gafgyt*' attack types, i.e., *gafgyt_udp*, *gafgyt_combo*, *gafgyt_scan*, and *gafgyt_junk,* are moderately frequent, occupying the middle ground between common *Mirai* types and *benign* traffic. This bias is a key characteristic of the dataset. The prevalence of some *Mirai* attack varieties implies that they are common attacks in test or real-world environments, and the absence of benign traffic promotes distrust of the dataset design, potentially skewing towards evil deeds.

From the modeling perspective, this imbalance is undesirable while training network security models. Models might become biased toward the more frequent attack types and thus reduce their effectiveness when identifying less frequent attacks or correctly distinguishing benign traffic. To avoid this flaw, future research should endeavor to apply data balancing strategies, e.g., resampling, class weighting, or synthetic data generation, to improve model generalization and offer effective detection across all types of traffic.

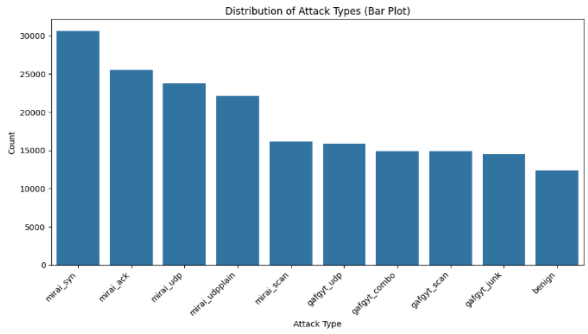

Figure 1: Distribution of Attack Types (Bar Plot)

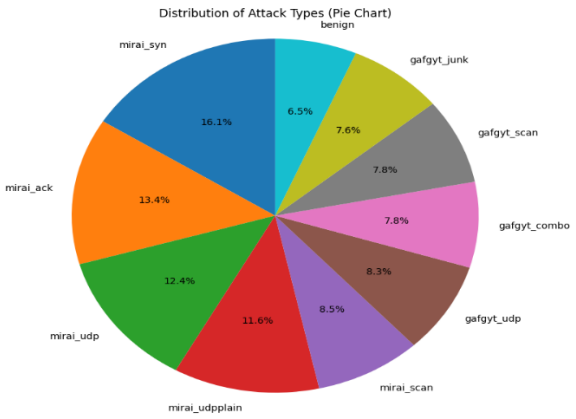

Figure 2: Distribution of Attack Types (Pie Chart).

Autoencoder

An autoencoder (AE) is a nonlinear unsupervised learning model that is trained to minimize the error between the input and output sets, typically using a subset of the data, such as the normal set only (Alaghbari *et al.,* 2023). AE consists of an encoder that transforms the input data into a compressed form, known as bottleneck features or latent space, and a decoder that transforms the compressed data back into the original data. AE is used to detect anomalies based on reconstruction error (RE) calculated from the difference between the original and reconstructed samples. To distinguish between normal and anomalous data, it usually produces higher RE for anomalous samples than for normal samples (Alaghbari *et al.,* 2023). In addition to that, it can be used as a feature extraction technique, where the compressed data is used for other purposes such as classification (Yeom *et al.,* 2020). Auto-encoder comprises two portions, the encoder and decoder. In other research, there are three components, and the third component is a middleware between both, known as code.

Encoders: The input data is compressed and represented in a smaller dimension by the encoder layer, which makes the compressed data appear to be the original data, but it is not.

Code: An encoder, also known as a bottleneck layer and denoted by the letter *z* in architecture, maps input space into lower lower-dimensional latent space. It now represents unsupervised data in a lower-dimensional manner. The component known as code is what the decoder receives as compressed input.

Decoder: The decoder extracts the original data of the same dimension from the encoded data. When the data is moved from the lower latent space by the decoder, the dimensionality of the output *x* bar matches output *x* in the reconstruction stage. If we consider it in terms of data compression, there is lossless compression, but auto-encoders use lossy compression involves reducing and then uncompressing the input. It requires an effort to be close to the input when it is uncompressed, but the outcome is different. (Agrawal, 2022).

The autoencoder-based model in Figure 1 was designed for detecting network traffic anomalies. It comprised the following layers: (i) Input, (ii) Encoder, (iii) Bottleneck, (iv) Decoder.

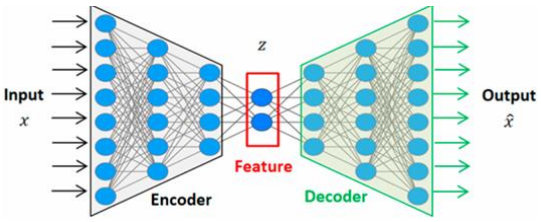

Figure 3: Auto-encoders Architecture Adopted from (Alaghbari *et al.,* 2023).

Autoencoder Proposed Architecture

The autoencoder used for anomaly detection consists of six main components: Input Layer, Encoder, Bottleneck, Decoder, Reconstructed Output, and Anomaly Detection Module. The input layer receives network traffic data *x* and feeds it into a series of fully connected layers of the encoder that sequentially compact the representation of data into the low-dimensional latent space, also known as the bottleneck. This layer preserves the majority of the significant features of normal traffic patterns and excludes noise and redundancy.

The decoder is also a reversed architecture of the encoder, transforming the input data $\hat{x}$ back from the latent space. The model is trained to minimize the reconstruction loss, typically in

Science World Journal Vol. 20(No 4) 2025 https://dx.doi.org/10.4314/swj.v20i4.8
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

terms of Mean Squared Error (MSE) between reconstructed output and input:

$$L = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \quad (1)$$

Training Process

The process of learning includes training the autoencoder with normal traffic data alone such that the model acquires knowledge of normal network behavior. Since the autoencoder learns only normal patterns, it is kept aware of nonconforming deviations from learned knowledge. The parameters of the model are optimized using the Adam optimizer with learning rate tuned for convergence stability. Training continues until reconstruction loss is stabilized, indicating that the model has learned the normal distribution of data.

Threshold for Anomaly Determination

After training, the model calculates the reconstruction error per data point. Data points that produce higher-than-normal reconstruction errors are flagged as potential anomalies. The threshold for anomaly detection is statistically determined from the distribution of the reconstruction errors over the training set. That is, a cutoff value is chosen by one of the following methods:

1) **Mean and Standard Deviation Rule**:

$$T = \mu + k\sigma \quad (2)$$

where μ and σ are the mean and standard deviation of reconstruction errors from normal data, and (k) is an empirically chosen factor (commonly between 2 and 3).

2) **Percentile-Based Thresholding:** Selecting a cutoff based on the 95th or 99th percentile of the error distribution.

The data point is classified as anomalous if the reconstruction error $E = |x - \hat{x}|$ surpasses the threshold T. This method guarantees an objective, data-driven method of differentiating between typical and anomalous patterns.

The trained autoencoder is used in deployment to process newly received IoT traffic data. The reconstruction error is calculated in real time by the system. The behavior is categorized as normal if the error stays below the threshold and as anomalous otherwise. Without the need for manual labeling, this procedure allows for real-time, unsupervised anomaly detection that adjusts to various IoT environments.

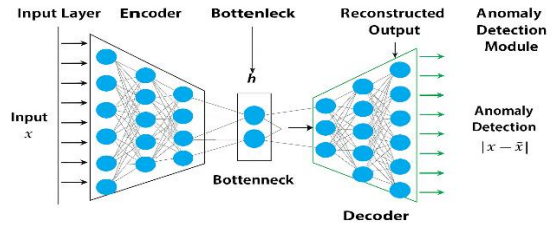

Figure 4: Proposed Architecture

Mathematical Formulation of Model

Let's denote the following variables:

- **X**: Input data, representing network traffic data samples.
- **X*i***: A specific data sample in the dataset **X**
- **H**: Hidden layer representation in the autoencoder.
- **H*i***: Hidden layer representation for data sample **X*i***.
- **D:** Output reconstruction of the autoencoder.
- **D*i*:** Output reconstruction for data sample **X*i.***
- **E**: Reconstruction error.
- **E*i*:** Reconstruction error for data sample **X*i.***
- **N:** Number of input features (dimensions) in each data sample.
- **M:** Number of neurons in the hidden layer of the autoencoder.

Data Preprocessing:

Normalize the input data **(X)** to have a zero mean and unit variance.

Autoencoder Architecture:

The autoencoder includes a hidden layer with M neurons, an output layer with N neurons, and an input layer with N neurons. (Reconstruction layer).

Encoding:

The encoding operation from input (**X***i*) to hidden representation (**H***i*) is given by:

$$\mathbf{H}i = f_{encode}(\mathbf{W}_{encode} * \mathbf{X}i + \mathbf{b}_{encode}) \quad (3)$$

where $\mathbf{W}_{encode}$ is the weight matrix and b_{encode} is the bias vector for the encoding operation, and f_{encode} is the activation function

Decoding:

The decoding operation from hidden representation H*i* to reconstruction D*i* is given by:

$$\mathbf{D}i = f_{decode}(\mathbf{W}_{\text{decode}} * \mathbf{H}i + \mathbf{b}_{\text{decode}}) \quad (4)$$

where $\mathbf{W}_{\text{decode}}$ is the weight matrix and $\mathbf{b}_{\text{decode}}$ is the bias vector for the decoding operation, and decode f_{decode} *is the activation function.*

Reconstruction Error:

Calculate the reconstruction error E*i* for data sample (**X***i*) as the difference between (**X***i*) and its corresponding reconstruction (**D***i*):

$$\mathbf{E}i = \mathbf{X}i - \mathbf{D}i \quad (5)$$

Anomaly Detection:

Define a threshold ($\boldsymbol{\tau}$) for anomaly detection. Data samples with reconstruction errors (**E***i*) exceeding this threshold are considered anomalies.

Training:

The autoencoder is trained to minimize the mean squared error (MSE) between the input data (**X***i*) and its reconstruction (**D***i*)

$$\boldsymbol{MSE} = \frac{1}{N}\sum_{j=1}^{N}(Xi,j - Di,j)^2 \quad (6)$$

The optimization process adjusts the weights and biases ($\boldsymbol{W}_{\text{encode}}$, $\boldsymbol{W}_{\text{decode}}$, $\boldsymbol{b}_{\text{encode}}$, $\boldsymbol{b}_{\text{decode}}$) to minimize the reconstruction error.

Science World Journal Vol. 20(No 4) 2025 https://dx.doi.org/10.4314/swj.v20i4.8
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

RESULTS AND DISCUSSION

Here, the study results are discussed and presented. Results are depicted in the following tables and figures. For accurate and reliable predictions in deep learning, particularly with deep autoencoders, we must evaluate the model using important measures of performance. These include Accuracy, Precision, Recall, F1-Score, and ROC Curves.

Table 1: Confusion Matrix's Predicted and Actual Class Parameter

Actual Class	Predicted Class	TP	FP	FN	TN
benign	benign	2422	2	3	35750
gafgyt_combo	gafgyt_combo	2810	151	63	35066
gafgyt_junk	gafgyt_junk	2762	151	60	35066
gafgyt_scan	gafgyt_scan	3074	9	2	35054
gafgyt_udp	gafgyt_udp	3164	3	2	35043
mirai_ack	mirai_ack	5058	14	7	33133
mirai_scan	mirai_scan	3240	1	0	34971
mirai_syn	mirai_syn	6185	0	0	31996
mirai_udp	mirai_udp	4746	17	15	33392
mirai_udpplain	mirai_udpplain	4466	0	0	33650

True Positives (TP): These are the correctly predicted positives as normal traffic, which means that the value of the actual class is normal, and the value of the predicted class is also normal traffic.
True Negatives (TN): These are the correctly predicted as an attack, which means that the value of the actual class attack is equal to the value of the predicted class attack.
False Positives: In our result, this shows the actual class is abnormal traffic/malware, but the classifier predicted it as normal traffic
False Negatives: The actual class is normal traffic, but the classifier predicts it as an attack.

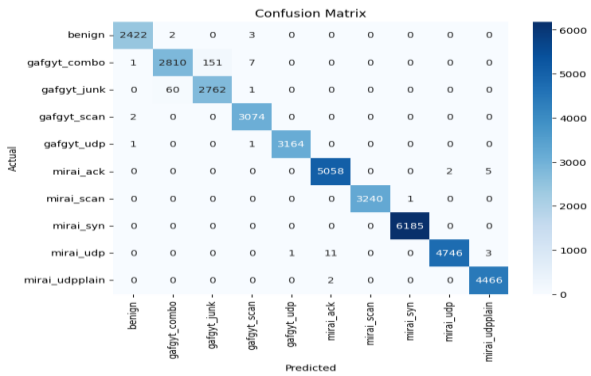

Figure 5: Confusion Matrix

Figure 4 shows how well the Autoencoder-based malware detection system performs using a confusion matrix. The matrix compares what the model predicted with the actual malware types. Each row represents the true class, and each column represents the predicted class. The darker the color in a cell, the more often that combination occurred. The diagonal cells show correct predictions, while the off-diagonal ones show where the model got it wrong. By looking at this, we can see which malware types the model identifies accurately and which ones it tends to confuse. Overall accuracy is determined by dividing the number of correct predictions by the total cases.

Table 2: Performance measures and the percentages of the model

	Class	Accuracy	Precision	Recall	F1-Score
0	0	0.999764	0.998351	0.997940	0.998145
1	1	0.994212	0.978412	0.946447	0.962164
2	2	0.994448	0.948163	0.978392	0.963040
3	3	0.999633	0.996111	0.999350	0.997728
4	4	0.999921	0.999684	0.999368	0.999526
5	5	0.999476	0.997436	0.998618	0.998027
6	6	0.999974	1.000000	0.999691	0.999846
7	7	0.999974	0.999838	1.000000	0.999919
8	8	0.999555	0.999579	0.996849	0.998212
9	9	0.999738	0.998212	0.999552	0.998882

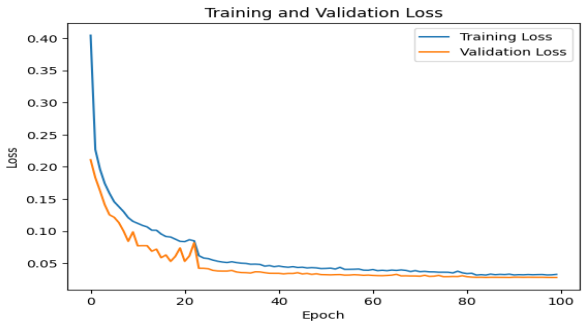

Figure 5A: Loss Comparison of AutoEncoder-Based Models.

Science World Journal Vol. 20(No 4) 2025 https://dx.doi.org/10.4314/swj.v20i4.8
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

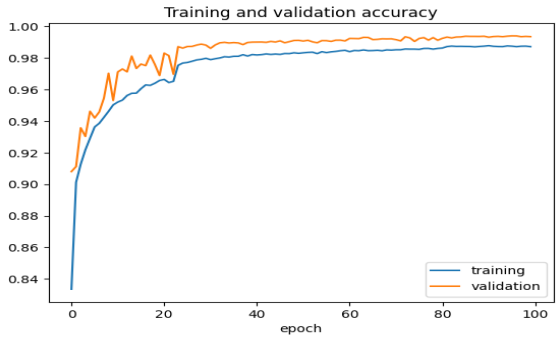

Figure 6: Accuracy Comparison of AutoEncoder-Based Models.

In Figures 5 and 6, the first plot shows both the training and testing sets' accuracy values. The accuracy values are displayed on the y-axis, and the epoch number is displayed on the x-axis. The accuracy is shown by the blue line. Values for the training set, while the orange line represents the accuracy values for the testing set. The legend indicates which line represents which set. The second plot shows the loss values for both the training and testing sets. The y-axis represents the loss values, while the x-axis represents the epoch number. The blue line represents the loss values for the training set, while the orange line represents the loss values for the testing set. The legend indicates which line represents which set. it is also noted that the model's accuracy is increasing with each epoch, and the loss is decreasing, which indicates that the model is learning from the data and is not overfitting. It is also important to note that the accuracy values for the testing set are not too far behind the accuracy values for the training set, which indicates that the model is generalizing well and can make accurate predictions on new, unseen data.

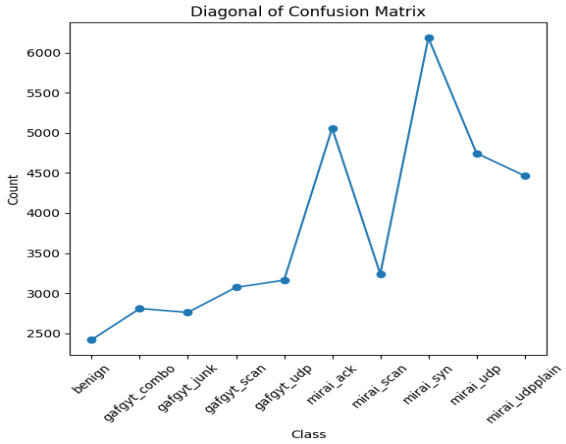

Figure 7: Diagonal of the Confusion matrix
In Figure 7 showing the number of data points properly separated by an autoencoder-based model is represented by a diagonal of the confusion matrix. A higher value of the diagonal indicates that the model can correctly classify more data points.

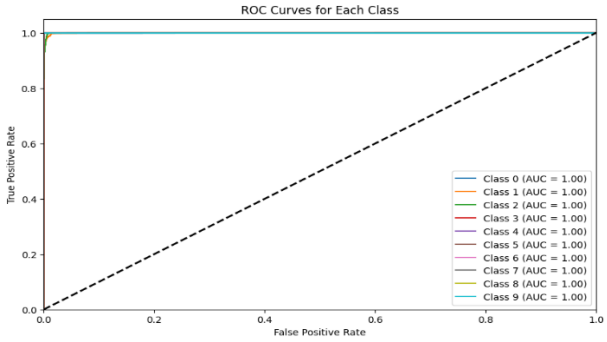

Figure 8: ROC Curves for Each Class

In Figure 8, the graph visualizes ROC (Receiver Operating Characteristic) curves for each class in an instance containing several classes. The effectiveness of a binary classification model is evaluated using ROC curves at various classification thresholds. Our model generates the AUC (area under the ROC curve) score (AUCAE = 1.00) with a very high true positive rate and a low false-positive rate, exhibiting its great performance.

Autoencoder-Based Classification Model Prediction
This Autoencoder-Based Classification Model Prediction Report for Excel File Data summarizes the predictions made by a trained classification model based on an autoencoder on imported data from the Excel file '/content/drive/MyDrive/network/gafgyt.scan_datase.xlsx'.
The model, designed to identify types of network traffic, analyzed the input data and produced the following predicted class labels in Figure 9:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739:
  warnings.warn(
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
Predicted class labels for the data in the Excel file:
['mirai_udpplain' 'mirai_udpplain' 'mirai_syn' 'mirai_udpplain'
 'mirai_udpplain' 'mirai_udpplain' 'mirai_udpplain' 'mirai_udpplain'
 'mirai_udpplain' 'mirai_udpplain' 'mirai_udpplain' 'mirai_udpplain'
 'mirai_udpplain' 'mirai_udpplain']
```

Figure 9: Predicted Class Label

From these correlations, it can be seen that most of the traffic within the dataset has been labeled as 'mirai_udpplain', which is generally linked with the Mirai botnet that utilizes UDP flooding to flood network services. A single data point was labeled as 'mirai_syn', which indicates the existence of SYN flood behavior, one type of attack that targets draining server resources.
Presence of 'mirai_udpplain' indicates likely UDP flood attack signatures in the respective traffic. Existence of 'mirai_syn' would indicate a mix of TCP SYN flood signatures, which could be used as one of the multi-vector attack approach vectors. This signature pattern may suggest that the network traffic in question is made up of malicious botnet operations and could require further analysis or protection measures.

Conclusion
This study developed an autoencoder-based model for detecting network traffic anomalies and achieved an impressive accuracy of 99%. The results show that the model can effectively identify

unusual network behavior, making it a promising tool for detecting potential cyberattacks.

However, there are a few limitations to note. The model was tested on a specific dataset, which means its performance in real-world networks may vary. Also, training deep autoencoders requires high computational power, which might make it difficult to use on small or low-resource devices.

In the future, it would be valuable to test this model on real-world IoT systems and across multiple datasets to confirm its reliability. Improving the model to handle live, streaming data for real-time anomaly detection and making it more lightweight for use on edge devices are also important directions for further research.

REFERENCE

Agrawal, R. (2022, January 10). *Complete guide to anomaly detection with autoencoders using TensorFlow*. Analytics Vidhya**.** https://www.analyticsvidhya.com/blog/2022/01/complete-guide-to-anomaly-detection-with-autoencoders-using-tensorflow/.

Alaghbari, K. A., Lim, H. S., Saad, M. H. M., & Yong, Y. S. (2023). Deep autoencoder-based integrated model for anomaly detection and efficient feature extraction in iot networks. *IoT*, *4*(3), 345-365.

Alaghbari, K. A., Saad, M. H. M., Hussain, A., & Alam, M. R. (2022). Activities recognition, anomaly detection and next activity prediction based on neural networks in smart homes. *IEEE Access*, *10*, 28219-28232.

Ali, S., Ghazal, R., Qadeer, N., Saidani, O., Alhayan, F., Masood, A., ... & Gupta, D. (2024). A novel approach of botnet detection using hybrid deep learning for enhancing security in IoT networks. *Alexandria Engineering Journal*, *103*, 88-97.

Al-Qudah, M., & AlMahamid, F. (2025, April). A Multi-Step Comparative Framework for Anomaly Detection in IoT Data Streams. In *2025 International Conference on New Trends in Computing Sciences (ICTCS)* (pp. 432-439). IEEE.

Ayad, A. G., El-Gayar, M. M., Hikal, N. A., & Sakr, N. A. (2024). Efficient Real-Time anomaly detection in IoT networks using One-Class autoencoder and deep neural network. *Electronics*, *14*(1), 104.

Qiu, K., Yan, M., Luo, T., & Chen, F. (2025). FedAware: a distributed IoT intrusion detection method based on fractal shrinking autoencoder. *Journal of King Saud University Computer and Information Sciences*, *37*(7), 1-21.

Rhachi, H., Balboul, Y., & Bouayad, A. (2025). Enhanced Anomaly Detection in IoT Networks Using Deep Autoencoders with Feature Selection Techniques. *Sensors*, *25*(10), 3150.

Shah, K., Sarwar, M., & Abdeljawad, T. (2024). A comprehensive mathematical analysis of fractal–fractional order nonlinear re-infection model. *Alexandria Engineering Journal*, *103*, 353-365.

Somma, M. (2025). Hybrid Temporal Differential Consistency Autoencoder for Efficient and Sustainable Anomaly Detection in Cyber-Physical Systems. *arXiv preprint arXiv:2504.06320*.

Torabi, H., Mirtaheri, S. L., & Greco, S. (2023). Practical autoencoder based anomaly detection by using vector reconstruction error. *Cybersecurity*, *6*(1), 1.

Yeom, S., Choi, C., & Kim, K. (2020, September). AutoEncoder based feature extraction for multi-malicious traffic classification. In *The 9th International Conference on Smart Media and Applications* (pp. 285-287).