

Reconstruction Based Unsupervised Intrusion Detection Using Autoencoder Latent Representation Analysis

Somnath Koley¹, Somya Das²

¹Assistant Professor, Institute of Advance Education and Research

²Student, Institute of Advance Education and Research

Abstract—The rapid growth of network-based applications and digital communication has made cyber security threats both more frequent and significantly more complex. Intrusion Detection Systems (IDS) play a critical role in identifying malicious activity and protecting network infrastructure from unauthorized access and evolving cyber-attacks.

However, traditional signature-based methods often struggle to detect previously unseen attack patterns, making anomaly-based approaches increasingly important. This paper proposes a fully unsupervised deep autoencoder-based intrusion detection framework using the NSL-KDD dataset.

The model was trained exclusively on legitimate network traffic to learn compressed latent representations of normal behavioural patterns without relying on labelled attack data. Suspicious traffic was identified through reconstruction error analysis. This framework goes beyond basic classification, incorporating statistical thresholding, sensitivity analysis, latent-space visualization, and clustering validation.

To examine how the autoencoder organized behaviour internally, Principal Component Analysis (PCA), K-Means clustering, and silhouette score analysis was applied to the learned latent feature space. Experimental evaluation demonstrated strong anomaly detection capability, achieving high precision, stable F1-score performance, and strong ROC-AUC and AUPRC values. The results suggest that the system successfully learned meaningful representations of legitimate traffic while effectively distinguishing anomalous behaviour.

Index Terms—Anomaly Detection, Cybersecurity, Deep Autoencoder, Intrusion Detection System, K-Means Clustering, Latent Representation Learning, Network Traffic Analysis, Principal Component Analysis, Reconstruction-Based Learning, Threshold Sensitivity Analysis, Unsupervised Learning.

I. INTRODUCTION

Modern computer networks generate enormous volumes of traffic every second, making cyber security threats increasingly difficult to monitor and contain. From denial-of-service (DoS) attacks and probing attempts to unauthorized access and privilege escalation exploits, malicious activities continue to evolve in both scale and sophistication. These attacks can lead to severe consequences, including data breaches, financial loss, and service disruption. As digital infrastructure becomes more interconnected, the need for intelligent and adaptive network security systems has become critical.

Intrusion Detection Systems (IDS) are designed to monitor network activity and identify suspicious behaviour. Traditional IDS approaches largely depend on signature-based detection, where incoming traffic is compared against predefined attack patterns. Although effective against known threats, signature-based systems often struggle to identify zero-day attacks and previously unseen attack variants. Their dependence on manually updated signatures limits their adaptability in rapidly evolving environments.

To address these limitations, anomaly-based intrusion detection techniques are increasingly gaining attention. Unlike signature-based methods, anomaly detection systems attempt to model normal network behaviour and identify deviations from expected patterns. This makes them particularly useful for detecting unknown or evolving attacks.

Recent advances in deep learning have further improved the potential of these systems. Among various architectures, autoencoders have shown strong capability for unsupervised anomaly detection because they can learn compressed representations of normal data and reconstruct behavioural patterns

efficiently. When trained exclusively on legitimate traffic, autoencoders typically produce higher reconstruction errors for anomalous samples, making reconstruction error analysis an effective mechanism for detection.

In this work, a fully unsupervised autoencoder-based framework is implemented using the NSL-KDD dataset. While the dataset is a controlled simulation, the focus remains on learning the structural characteristics of legitimate behavior to create a system that can adapt more effectively to unseen anomalies.

The major contributions of this work are summarized below:

- Development of a fully unsupervised autoencoder-based intrusion detection framework.
- Reconstruction-error-based identification of anomalous network traffic using MSE.
- Latent-space analysis and representation visualization using PCA.
- Cluster analysis of learned representations using K-Means and Silhouette scores.
- Comprehensive evaluation using metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and AUPRC.
- Behavioural analysis of reconstruction patterns across multiple attack categories within the NSL-KDD dataset.

II. LITERATURE REVIEW

Intrusion Detection Systems (IDS) play an important role in modern cyber security by monitoring network traffic for suspicious activity. Broadly, these approaches are divided into signature-based and anomaly-based detection. Signature-based IDS models identify attacks by comparing incoming traffic against predefined malicious patterns. These systems are highly effective against identified threats but struggle with zero-day attacks and evolving variants. As threats become more adaptive, the limitations of purely signature-based systems have become increasingly noticeable.

This has led to growing interest in anomaly-based techniques. Instead of relying on known signatures, these systems learn normal network behaviour and identify deviations. Machine learning approaches have been widely explored here due to their ability to

uncover hidden structures within complex traffic data.

Supervised models like Decision Trees, Support Vector Machines (SVM), and Random Forests have demonstrated strong performance. However, these approaches rely heavily on labelled datasets containing both legitimate and malicious traffic. In practice, obtaining high-quality labelled data is often expensive, time-consuming, and difficult to scale.

To overcome these limitations, unsupervised deep learning approaches have gained significant attention. Among them, autoencoders have emerged as a widely used technique. Autoencoders learn compact latent representations of normal traffic and reconstruct the input data through an encoder-decoder architecture.

The foundational work of Geoffrey Hinton and Ruslan Salakhutdinov demonstrated that deep autoencoders could learn meaningful low-dimensional feature representations through dimensionality reduction. Their research significantly influenced the development of representation learning in network security.

Several studies have applied autoencoder-based systems using benchmark datasets such as KDD Cup 1999, NSL-KDD, CICIDS2017, and UNSW-NB15. Most train exclusively on normal samples and classify anomalies using reconstruction error thresholds. Compared to traditional systems, deep learning-based IDS models have shown stronger adaptability toward unseen attack patterns.

Recent studies also focused on improving interpretability through latent-space analysis. Methods such as PCA, K-Means, and silhouette score analysis have been used to examine the structural organization of learned features. Despite this progress, challenges like threshold selection, false positive reduction, and generalization to modern attacks remain active areas of research.

III. METHODOLOGY

A. Development Environment

The proposed intrusion detection system was implemented using Python and machine learning and data analysis libraries. Pandas and NumPy were used for data manipulation and numerical computation. Scikit-learn was used for preprocessing, feature scaling, clustering, and evaluation metrics.

TensorFlow and Keras were used for designing and training the deep autoencoder architecture. Matplotlib and Seaborn were used for visualization and graphical analysis of model performance.

B. System Architecture

The overall workflow of the proposed intrusion detection system is illustrated below. The system consists of multiple stages including data preprocessing, unsupervised autoencoder training, reconstruction-based anomaly detection, latent-space analysis, clustering validation, and performance evaluation. The architecture was designed to enable fully unsupervised intrusion detection while also providing behavioural interpretation and latent feature analysis of network traffic patterns.

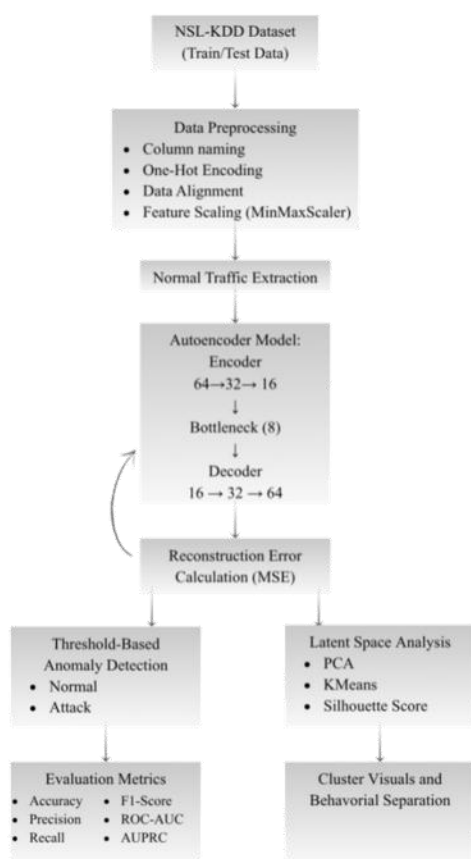


Fig. 3.1 System Architecture

C. Dataset Collection

The proposed intrusion detection system was developed using the NSL-KDD dataset, a refined version of the KDD Cup 1999 dataset commonly used for the evaluation of intrusion detection systems. [1]

The dataset contains labeled records of normal and malicious network traffic categorized into multiple attack classes.

Each row in the dataset represents a single network connection and describes the behavior of a network interaction through communication characteristics and connection statistics. The traffic records are labeled as either normal or attack traffic. The attack instances are further categorized into four major classes:

- Denial of Service (DoS) Attacks
These attacks attempt to overwhelm system resources or network services, making them unavailable to legitimate users.
- Probe Attacks These attacks involve scanning and information gathering activities intended to identify vulnerabilities, open ports, or network weaknesses.
- Remote to Local (R2L) Attacks These attacks occur when an external attacker gains unauthorized access to a local system without prior account access.
- User to Root (U2R) Attacks These attacks involve privilege escalation, where a normal user attempts to gain administrative or root-level access to the system.

The NSL-KDD dataset consists of 41 features that describe various properties of network traffic. These features can be broadly classified into the following categories:

1. Basic Features

These features describe fundamental properties of individual network connections and direct traffic behavior.

Examples include:

- Protocol type
- Service
- Duration

2. Content Features

These features assist in detecting unauthorized access attempts and exploit-based attacks through packet content and login-related activity.

Examples include:

- Failed login attempts
- Shell access
- Login activity

3. Traffic Features

Traffic-based features capture statistical patterns and connection behavior over time, helping identify suspicious activities such as scanning, flooding, and coordinated attacks.

Examples include:

- Connection frequency
- Count of related connections
- Repeated requests

4. Label Features

These indicate whether a connection belongs to normal traffic or a specific attack category.

The NSL-KDD dataset is publicly available and widely used in cybersecurity research for intrusion detection, anomaly detection, and network traffic analysis. One of the major advantages of the dataset is its structured format and benchmark consistency, which make it suitable for experimentation and performance evaluation of machine learning models. However, the dataset also has certain limitations. Since it was generated in a controlled simulation environment, it does not fully represent the complexity and unpredictability of modern real-world network traffic. Additionally, some attack patterns reflected in the dataset are relatively outdated compared to contemporary cybersecurity threats. Despite these limitations, NSL-KDD remains a widely accepted benchmark dataset because of its accessibility, computational efficiency, and suitability for academic experimentation.

D. Data Preprocessing

The raw NSL-KDD dataset was transformed into a structured format suitable for training and evaluation of the proposed autoencoder model.

Initially, the training and testing datasets were loaded separately using Pandas. Column names corresponding to the NSL-KDD feature descriptions were manually assigned to simplify feature handling during preprocessing.

1. Binary Label Transformation

The original NSL-KDD dataset contains multiple attack labels representing different categories of malicious activity. These labels were converted into binary classes, where normal traffic was labelled as 0 and attack traffic was labelled as 1.

This binary transformation simplified the intrusion detection task by enabling the model to distinguish

between legitimate network traffic and anomalous behaviour.

2. Categorical Feature Encoding

The NSL-KDD dataset contains categorical attributes such as protocol type, service, and flag. Since neural networks cannot directly process textual categorical values, One-Hot Encoding was applied using the `get_dummies()` function. This transformed each categorical value into a separate binary feature representation.

To ensure consistency between training and testing datasets after encoding, both datasets were aligned using feature matching operations. Missing columns generated during alignment were filled with zero values to maintain structural uniformity between training and testing samples.

3. Feature Selection

Before model training, the following attributes were excluded from the dataset:

- label
- binary_label
- difficulty

The label and binary_label attributes were removed because they represent target classification information rather than actual network traffic characteristics. Including these attributes during training would introduce data leakage and compromise the integrity of anomaly detection.

The difficulty attribute was excluded because it serves as a dataset-specific evaluation parameter and does not contribute robust behavioral information related to network traffic analysis.

The remaining processed features were retained as input attributes for the proposed autoencoder model.

4. Feature Scaling

Feature normalization was performed using `MinMaxScaler`, which scales numerical values into a range between 0 and 1.

$$x' = (x - x_{\min}) / (x_{\max} - x_{\min})$$

where:

- x represents the original value
- $x_{\min}$ represents the minimum feature value
- $x_{\max}$ represents the maximum feature value
- x' represents the normalized value

The scaler was fitted only on the training dataset and subsequently applied to the testing dataset to prevent data leakage during evaluation.

Normalization was necessary because the dataset contains features with significantly different numerical ranges. Scaling ensured stable neural network training and compatibility with the sigmoid activation function used in the output layer of the autoencoder.

5. Extraction of Normal Training Samples

Since the proposed system follows an unsupervised anomaly detection approach, the training dataset used consisted exclusively of normal traffic samples extracted from the training dataset. This enabled the model to learn the normal behavioural distribution of network traffic and identify anomalous patterns based on deviations from learned representations.

Attack samples were excluded during model training in order to preserve the fully unsupervised anomaly detection setting. The testing dataset, however, contained both normal and anomalous traffic samples to evaluate the anomaly detection capability of the proposed system under realistic intrusion scenarios.

E. Proposed Autoencoder Architecture

The proposed intrusion detection system was implemented using a deep, fully connected autoencoder architecture developed using TensorFlow and Keras [2]. The architecture was designed to derive compressed latent representations of normal network traffic and reconstruct the original input patterns with minimal reconstruction error.

The overall architecture followed the sequence:

Input → Encoder → Bottleneck → Decoder → Output

1. Encoder Network

The encoder network received preprocessed network traffic data and progressively reduced the dimensionality of the input features through multiple dense layers.

The encoder architecture followed the structure:

Input → 64 → 32 → 16 → 8

ReLU activation functions were used in each hidden layer to introduce non-linearity and improve representation learning capability.

Dropout regularization with a dropout rate of 0.2 was applied after each hidden layer to reduce overfitting and improve generalization performance.

2. Bottleneck Layer

The bottleneck layer represented the compressed latent representation of network traffic behaviour and consisted of 8 neurons.

By reducing the dimensionality of the original feature space, the model was forced to prioritize the underlying structural characteristics of normal traffic instead of memorizing redundant information.

3. Decoder Network

The decoder network reconstructed the compressed latent representations back into the original feature dimensions.

The decoder architecture followed the structure:

8 → 16 → 32 → 64 → Output

The final output layer used a sigmoid activation function to produce normalized reconstructed outputs compatible with MinMax-scaled input features.

The difference between the reconstructed output and the original input was used to evaluate how effectively the model learned the underlying patterns of the input data.

F. Model Training and Regularization

1. Loss Function and Optimization

2. Early Stopping

Early stopping was implemented during training using validation loss monitoring.

The training process was configured with:

- patience = 3
- min_delta = 0.0001
- restore_best_weights = True

Training automatically terminated when the validation loss failed to improve significantly for three consecutive epochs. This reduced unnecessary training iterations and minimized the possibility of overfitting while preserving the best performing model weights.

The autoencoder model was trained for a maximum of 100 epochs using a batch size of 256. A validation split of 0.2 was used during training, and the training samples were shuffled before each epoch to improve learning generalization.

G. Anomaly Detection Mechanism

After training, reconstruction error values were calculated for each sample produced by the autoencoder using Mean Squared Error (MSE) between the original input features and reconstructed outputs.

Samples producing higher reconstruction errors indicated greater deviation from learned normal

traffic patterns and were therefore considered anomalous.

The anomaly threshold was calculated using the mean and standard deviation of reconstruction errors obtained from the training data. [3]

Threshold = $\mu + 3\sigma$

where:

- μ represents the mean reconstruction error
- σ represents the standard deviation of reconstruction error

Samples with reconstruction error values exceeding the calculated threshold were classified as anomalous, whereas samples below the threshold were classified as normal traffic.

Additionally, threshold sensitivity analysis was performed using percentile-based threshold variations ranging from 30 to 95 in order to study the impact of threshold selection on precision, recall, and F1-score performance.

H. Performance Evaluation Metrics

The performance of the proposed intrusion detection system was evaluated using multiple classification and anomaly detection metrics.

The evaluation metrics included:

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC score
- Area Under Precision-Recall Curve (AUPRC)

A confusion matrix and classification report were also generated to analyze classification performance across normal and anomalous traffic samples.

ROC and Precision-Recall curves were plotted to evaluate the discrimination capability of the proposed model across different threshold settings.

AUPRC was included because anomaly detection datasets often contain imbalanced class distributions, making it a more reliable assessment of the model's reliability than standard accuracy.

I. Latent Space and Clustering Analysis

Latent feature representations were extracted from the bottleneck layer of the trained autoencoder model for visualization and clustering analysis.

Principal Component Analysis (PCA) was applied to reduce the 8-dimensional latent vectors into a 2-dimensional representation for visualization

purposes. This enabled graphical interpretation of traffic behavior patterns learned by the autoencoder.

K-Means clustering was subsequently applied to the latent representations to analyze grouping behavior within the learned feature space.

To evaluate cluster quality and separation, silhouette score analysis [10] was performed for cluster values ranging from K=2 to K=14.

The latent-space analysis provided additional insight into the internal feature representations learned by the proposed autoencoder model. PCA visualization and clustering analysis demonstrated structural grouping behavior within the learned latent space, indicating that the model captured distinct behavioral patterns of network traffic beyond simple anomaly classification.

IV. RESULTS AND ANALYSIS

Performance analysis was conducted using classification metrics, reconstruction-error analysis, threshold sensitivity evaluation, and latent-space clustering analysis.

The results obtained from the proposed model demonstrate its capability to learn non-trivial representations of legitimate network traffic and effectively identify anomalous behaviour through reconstruction error deviations.

Additional analysis of latent representations and clustering behaviour further provided insight into the internal feature learning characteristics of the autoencoder architecture.

A. Performance Evaluation Results

The anomaly threshold was statistically derived from the reconstruction error distribution of normal training traffic using a three-standard-deviation criterion, resulting in a threshold value of 0.016. The proposed model achieved the following performance results:

- Accuracy: 87%
- Precision: 94%
- Recall: 82.41%
- F1 Score: 87.84%
- ROC AUC Score: 95.21%
- Area Under Precision-Recall Curve (AUPRC): 96.31%

Despite being trained exclusively on normal network traffic samples, the proposed autoencoder model

successfully identified anomalous traffic behavior with high precision and strong discriminative capability.

The high precision score indicates that the proposed model was highly effective in minimizing false positive predictions, meaning that most traffic samples classified as attacks were genuinely anomalous. The recall score demonstrates that the model was able to successfully identify a significant portion of malicious traffic instances.

The high Precision (94%) is particularly significant for practical deployment, as it minimizes 'alert fatigue' for security analysts by ensuring that flagged anomalies are highly likely to be actual threats.

The F1-score reflects a balanced tradeoff between precision and recall, indicating stable anomaly detection performance. Additionally, the high ROC-AUC and AUPRC values demonstrate strong discriminative capability between normal and anomalous traffic samples across varying threshold conditions.

Classification report:

Table 4.1 Classification report

Class	Precision	Recall	F1-Score	Support
Normal Traffic (0)	0.80	0.93	0.86	9711
Attack Traffic (1)	0.94	0.82	0.88	12833
Metric Type	Precision	Recall	F1-Score	Support
Accuracy	-	-	0.87	22544
Macro Average	0.87	0.88	0.87	22544
Weighted Average	0.88	0.87	0.87	22544

B. Confusion Matrix Analysis

The confusion matrix was generated to evaluate the classification performance of the proposed model on normal and anomalous traffic samples.

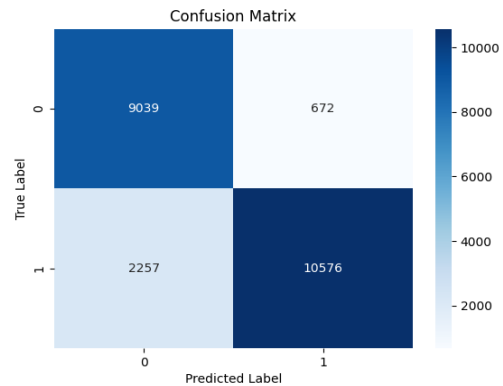


Fig. 4.1 Confusion Matrix

The model achieved strong detection performance for anomalous traffic while maintaining relatively low false positive predictions. A larger number of correctly classified attack samples indicated that the autoencoder successfully learned the behavioural characteristics of legitimate network traffic and identified deviations effectively.

Some false negatives were still observed, indicating that certain attack samples produced reconstruction patterns similar to normal traffic behaviour. This suggests that specific attack categories may share structural similarities with legitimate traffic, making anomaly separation more challenging.

C. Training and Validation Loss Analysis

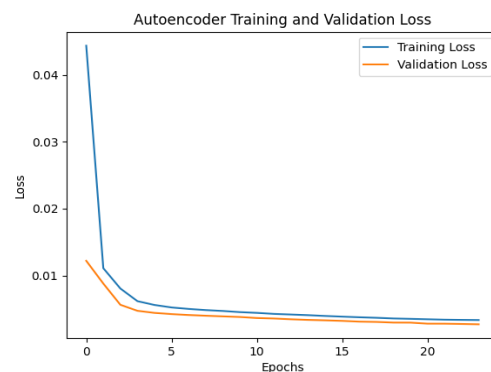


Fig. 4.2 Training and Validation Loss Analysis

During training, both training loss and validation loss decreased progressively across epochs, indicating successful learning of normal traffic patterns by the autoencoder model.

The validation loss closely followed the training loss without major divergence, suggesting that the proposed model generalized effectively and did not suffer from severe overfitting.

The lack of divergence between the training and validation curves confirms that the 0.2 dropout rate and early stopping strategy effectively prevented the model from over-fitting to the training set.

D. Reconstruction Error Distribution Analysis

Reconstruction error analysis was performed to study the separation between normal and anomalous traffic samples.

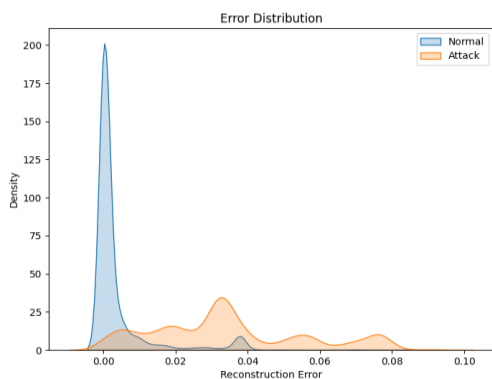


Fig. 4.3 KDE Plot of Reconstruction Error Distribution

The reconstruction error distributions showed that normal traffic samples generally produced lower reconstruction errors, whereas attack traffic generated significantly higher reconstruction errors.

The KDE plot further illustrated the separation between the reconstruction error distributions of normal and attack samples, validating the effectiveness of reconstruction-based anomaly detection.

E. ROC and Precision-Recall Curve Analysis

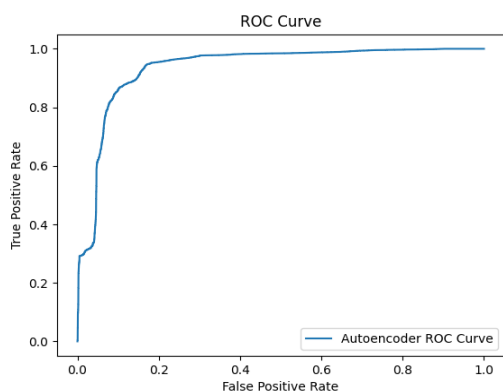


Fig. 4.4 ROC Curve of the proposed model

The ROC curve demonstrated strong classification capability across multiple threshold values, as reflected by the high ROC-AUC score of 95.21%.

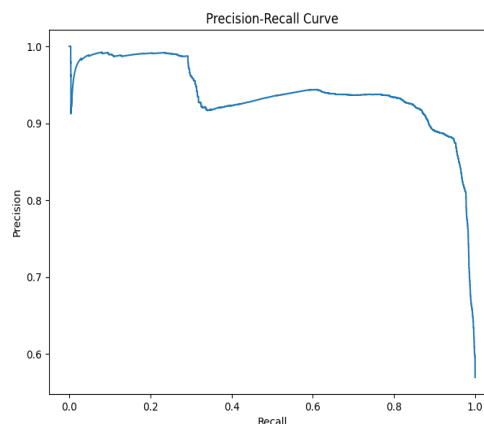


Fig. 4.5 Precision-Recall Curve

Similarly, the Precision-Recall curve showed stable precision-recall behaviour despite class imbalance within the dataset. The high AUPRC score indicated that the proposed model maintained strong anomaly detection capability while minimizing incorrect attack predictions.

Precision-Recall analysis was particularly important because intrusion detection datasets often contain imbalanced distributions between normal and attack traffic samples.

F. Threshold Sensitivity Analysis

Threshold sensitivity analysis was performed using reconstruction-error percentiles ranging from 30 to 95.

Precision, recall, and F1-score were evaluated across different threshold values to study the impact of threshold selection on anomaly detection performance.

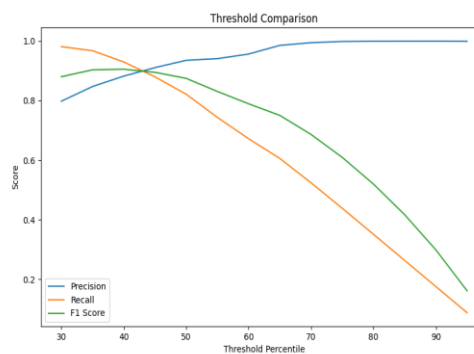


Fig. 4.6 Threshold Sensitivity Analysis Across Different Percentiles

Lower threshold values increased recall by classifying a larger number of samples as anomalous but also increased false positive predictions. Conversely, higher threshold values improved precision while reducing recall.

Table 4.2 Threshold Sensitivity Analysis

Threshold Percentile	Precision	Recall	F1-Score
30	0.7981	0.9815	0.8803
35	0.8461	0.9661	0.9021
40	0.8807	0.9283	0.9039
45	0.9127	0.8819	0.8970
50	0.9274	0.8146	0.8674
55	0.9365	0.7404	0.8270
60	0.9758	0.6857	0.8055
65	0.9911	0.6094	0.7548
70	0.9926	0.5231	0.6851
75	0.9927	0.4360	0.6059
80	0.9920	0.3486	0.5159
85	0.9935	0.2618	0.4144
90	0.9991	0.1756	0.2986
95	1.0000	0.0879	0.1616

The threshold sensitivity analysis demonstrated a clear tradeoff between precision and recall across different reconstruction-error thresholds. Lower threshold percentiles produced higher recall values by classifying a larger number of samples as anomalous, but also increased false positive predictions. Conversely, higher threshold percentiles significantly improved precision while reducing anomaly detection sensitivity.

The highest F1-score was observed around the 40th percentile threshold, achieving an optimal score of 90.39%, indicating a comparatively balanced trade-off between precision and recall under those threshold conditions. While lower percentiles offer nearly perfect recall, they compromise the system's reliability by introducing too much noise into the detection results.

The analysis highlights the importance of threshold selection in reconstruction-based anomaly detection systems.

G. Attack-Type Reconstruction Error Analysis

Average reconstruction error analysis was performed across different attack categories to study the behavioral sensitivity of the proposed model toward specific attack patterns.

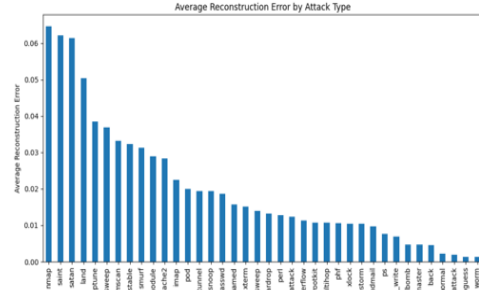


Fig. 4.7 Average Reconstruction Error Analysis

It is observed that 'nmap' and 'satan' attacks produced the highest errors, as their scanning-heavy nature is a departure by a big margin from the steady state traffic the model was trained on. Whereas attacks like 'back' are more deceptive, mimicking normal connection patterns and resulting in lower reconstruction errors. The variation in reconstruction error across attack categories demonstrated that different attack behaviors deviated from learned normal traffic patterns to varying degrees. Attack types producing higher reconstruction errors were structurally more distinguishable from legitimate traffic, whereas attacks with lower reconstruction errors exhibited partial similarity to normal behavioural patterns.

H. Latent Space Visualization and Clustering Analysis

Latent feature representations extracted from the bottleneck layer were analyzed using PCA-based visualization and K-Means clustering.

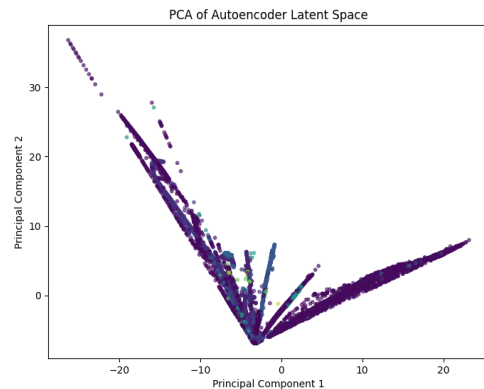


Fig. 4.8 PCA Visualization of Latent Representations by Attack Type

PCA projection of the latent vectors into two dimensions revealed visible grouping patterns between different traffic behaviours, indicating that the autoencoder learned meaningful compressed feature representations. K-Means clustering performed on the latent representations further demonstrated structural organization within the learned feature space.

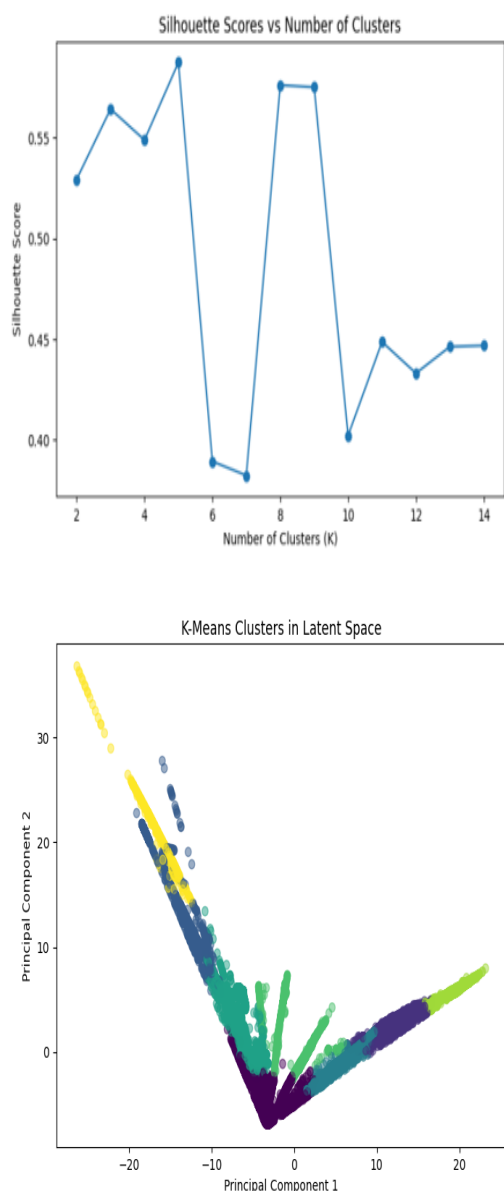


Fig. 4.9 K-Means Clustering in Latent Space
Silhouette score analysis was conducted for cluster values ranging from K=2 to K=14 to evaluate cluster separation quality.

Table 4.3 Latent Space Visualisation and Clustering Analysis

K Value	Silhouette Score
2	0.5293
3	0.5642
4	0.5489
5	0.5873
6	0.3894
7	0.3824
8	0.5761
9	0.5751
10	0.4022
11	0.4487
12	0.4331
13	0.4463
14	0.4468

The highest silhouette score was observed at K=5, indicating comparatively stronger cluster compactness and separation within the latent feature space at this cluster configuration. The variation in silhouette scores across different cluster values demonstrates the structural organization learned by the proposed autoencoder model within the latent representations.

Fig. 4.10 Silhouette Scores for Different Cluster Values

The clustering and latent-space visualization results indicate that the proposed autoencoder architecture learned salient compressed feature representations of network traffic behaviour. The observed structural separation within the latent space further validates the effectiveness of the bottleneck representations for unsupervised anomaly characterization and behavioural grouping.

V. CONCLUSION

This research implemented an autoencoder-based anomaly detection framework to address the limitations of traditional, signature-heavy security tools. By leveraging unsupervised deep learning on the NSL-KDD dataset, the model successfully learned compressed representations of legitimate network behaviour, allowing for the detection of anomalies through reconstruction error analysis.

The experimental results validate that modeling normalcy is a highly effective strategy for identifying malicious traffic. The high ROC-AUC and AUPRC scores demonstrate the model's discriminative power, while the latent-space analysis via PCA and K-Means confirms that the architecture did not merely memorize data but organized it into meaningful behavioural clusters. Specifically, the peak silhouette score at $K=5$ provides strong evidence that the model internally recognized the structural differences between normal traffic and the four primary attack categories.

While effective, the study highlights critical challenges in anomaly detection. The reliance on the NSL-KDD dataset means the findings are based on simulated environments that may not mirror the volatility of modern, live-streamed network traffic. Furthermore, the analysis of reconstruction errors by attack type revealed that deceptive attacks mimicking normal behaviour remain more difficult to isolate than high-intensity scanning or DoS attempts. Ultimately, this work proves that unsupervised autoencoders offer a scalable and adaptive foundation for next-generation IDS, especially in scenarios where labelled threat data is unavailable.

VI. FUTURE WORK

Although the proposed fully unsupervised autoencoder-based intrusion detection system demonstrated strong anomaly detection capability on the NSL-KDD dataset, several improvements and extensions can be explored in future work.

One important extension involves evaluating the proposed system on more modern and realistic cybersecurity datasets such as CICIDS2017 and UNSW-NB15. Since the NSL-KDD dataset represents older network traffic behaviour, testing the proposed model on contemporary datasets would improve evaluation reliability under modern attack scenarios.

Future work may also involve deploying the proposed intrusion detection framework in real-time network monitoring environments capable of analyzing live packet streams and dynamically identifying anomalous traffic behaviour.

The current system utilizes a statistical thresholding mechanism based on reconstruction error distribution. Future research may explore adaptive or

dynamic thresholding techniques capable of automatically adjusting anomaly sensitivity under changing traffic conditions.

More advanced deep learning architectures such as Variational Autoencoders (VAEs), LSTM-Autoencoders, Transformer-based models, and Graph Neural Networks (GNNs) may also be investigated to improve temporal feature learning and anomaly detection capability.

Additionally, future work may focus on improving explainability and interpretability of learned latent representations through Explainable Artificial Intelligence (XAI) techniques, enabling deeper understanding of anomaly behaviour and decision-making processes within intrusion detection systems.

ACKNOWLEDGMENT

I would like to sincerely thank Dr. Somnath Koley for valuable guidance, continuous support, and constructive feedback throughout the development of this research work. I also thank the Department of Data Science and the open-source research community whose tools and resources contributed significantly to this study.

REFERENCES

- [1] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA)*, Ottawa, ON, Canada, 2009, pp. 1-6.
- [2] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504-507, Jul. 2006.
- [3] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1-18, 2015.
- [4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

- [6] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, 2006.
- [7] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Machine Learning (ICML)*, Pittsburgh, PA, USA, 2006, pp. 233-240.
- [8] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symp. Mathematical Statistics and Probability*, Berkeley, CA, USA, 1967, pp. 281-297.
- [9] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53-65, 1987.
- [10] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2002.
- [11] F. Chollet et al., "Keras." [Online]. Available: Keras GitHub Repository
- [12] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: TensorFlow Official Website
- [13] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
- [14] W. McKinney, "Data structures for statistical computing in Python," in *Proc. 9th Python in Science Conf.*, Austin, TX, USA, 2010, pp. 56-61.