



WALLAGA UNIVERSITY

SCHOOL OF POSTGRADUATE STUDIES

**ANOMALY DETECTION IN CYBERSECURITY USING
DEEP LEARNING APPROACH**

DEPARTMENT OF COMPUTER SCIENCE

MSC. THESIS

BY: AKLILU BEDASSA HUNDE

May, 2026

Nekemte, Ethiopia



**WALLAGA UNIVERSITY
SCHOOL OF GRADUATE STUDIES**

**ANOMALY DETECTION IN CYBERSECURITY USING
DEEP LEARNING APPROACH**

**A thesis submitted to the department of computer science in partial
fulfillment for the degree of Master of Science in computer science**

BY: AKLILU BEDASSA HUNDE

Major Advisor: OBSA GILO WAKUMA (PhD)

May, 2026

Nekemte, Ethiopia



WALLAGA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

P.O. Box: 395, Nekemte, Ethiopia.

APPROVAL SHEET FOR SUBMITTING FINAL THESIS

As members of the Board of Examining of the Final MSc. thesis open defense, we certify that we have read and evaluated the thesis prepared by Aklilu Bedassa Hunde Under the Title “Anomaly Detection In Cybersecurity Using Deep Learning Approach” and recommend that the thesis be accepted as fulfilling the thesis requirement for the Degree of Master of Science in Computer Science

_____	_____	_____
Chairperson	Signature	Date

_____	_____	_____
Internal Examiner	Signature	Date

_____	_____	_____
External Examiner	Signature	Date

Final Approval and Acceptance

Thesis Approved by

_____	_____	_____
Department PGC	Signature	Date

_____	_____	_____
Dean of College	Signature	Date

Certification of the Final Thesis

I hereby certify that all the correction and recommendation suggested by the board of examiners are incorporated into the final thesis entitled “**Anomaly Detection in Cybersecurity Using Deep Learning Approach**” By **Aklilu Bedassa Hunde**

_____	_____	_____
Department PGC	Signature	Date

STATEMENT OF THE AUTHOR

I, Aklilu Bedassa Hunde, sincerely declare that the thesis titled “*Anomaly Detection in Cybersecurity Using Deep Learning Approach*” is my original work carried out under the supervision of Dr. OBSA.G. Throughout the research process including proposal writing, data collection, analysis, and final compilation I have strictly followed ethical standards of academic scholarship. Every source and scholarly material referenced in this work has been properly acknowledged and cited. I affirm that all references are accurately documented, and I have upheld principles of academic integrity by refraining from plagiarism, data fabrication, or misrepresentation of any information, data, or source. This study is submitted as part of the requirements for the **Master’s degree** at the School of Postgraduate Studies, Wallega University. I also confirm that this study has not been submitted elsewhere for any academic award. I am fully aware that any breach of these principles may lead to disciplinary actions by the University and potential legal consequences from original authors or institutions where appropriate permissions were not obtained.

Name: Aklilu Bedassa Hunde

Signature:  _____

Date: 19/5/2026 G.C

School/Department: Computer Science

College: Engineering And Technology

ACKNOWLEDGEMENTS

First and foremost, I want to sincerely thank the Almighty God for his blessing on the successful completion of this study. Next, I would like to convey my sincere gratitude to my adviser, OBSA.G (PHD) for unwavering commitment, helpful advice, and suggestions from the beginning of the project to its conclusion. Additionally, I am grateful to my family and friends specially (Mr. Laligo Legde) for their well-wishes, which helped me stay focused on this MSc. Journey & achieving my Future Goals. I am also gratefully acknowledging Wallega University for their cooperation to give this opportunity to pursue my Master's program.

DECLARATION

This is to certify that the thesis prepared by Aklilu Bedassa Hunde, entitled: “**Anomaly Detection in Cybersecurity Using Deep Learning Approach**” and submitted in partial fulfillment of the requirements for the Degree of Masters of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Name of Advisor: OBSA GILO WAKUMA (PhD)

Signature: _____



Date: 18/6/2026 G.C

ACRONYMS AND ABBREVIATIONS

– AD	Anomaly Detection
– ADWIN	Adaptive Windowing
– AE	Autoencoder
– AETs	Advanced Evasion Techniques
– APTs	Advanced Persistent Threats
– ARIMA	AutoRegressive Integrated Moving Average
– AUC	Area Under the Curve
– BiLSTM	Bidirectional Long Short-Term Memory
– Botnet	Robot Network (contextual term, not acronym)
– C2	Command-and-Control
– CAVs	Concept Activation Vectors
– CNN	Convolutional Neural Network
– CTGAN	Conditional Tabular Generative Adversarial Networks
– DAE	Denoising Autoencoder
– DBSCAN	Density-Based Spatial Clustering of Applications with Noise
– DDoS	Distributed Denial of Service
– DGAs	Domain Generation Algorithms
– DL	Deep Learning
– FPR	False Positive Rate
– GAN	Generative Adversarial Network
– GNN	Graph Neural Network
– GRU	Gated Recurrent Unit
– IDS	Intrusion Detection System
– IoT	Internet of Things
– IQR	Interquartile Range
– k-NN	k-Nearest Neighbors
– LIME	Local Interpretable Model-agnostic Explanations
– LSTM	Long Short-Term Memory

- **ML** Machine Learning
- **MITRE ATT&CK** Adversarial Tactics, Techniques, and Common Knowledge
- **NIDS** Network Intrusion Detection System
- **NIST** National Institute of Standards and Technology
- **NSL-KDD** Network Security Lab Knowledge Discovery and Data Mining
- **RNN** Recurrent Neural Network
- **SAE** Sparse Autoencoder
- **SHAP** SHapley Additive exPlanations
- **SMOTE** Synthetic Minority Oversampling Technique
- **SOC** Security Operations Center
- **SVM** Support Vector Machines
- **TLS** Transport Layer Security
- **VAE** Variational Autoencoder
- **XAI** Explainable Artificial Intelligence

TABLE OF CONTENTS

APPROVAL SHEET FOR SUBMITTING FINAL THESIS	I
STATEMENT OF THE AUTHOR	II
ACKNOWLEDGEMENTS	III
DECLARATION	IV
ACRONYMS AND ABBREVIATIONS	V
LIST OF TABLES	XI
LIST OF FIGURES	XII
ABSTRACT.....	XIII
CHAPTER ONE	1
1. INTRODUCTION	1
1.1. Background of the Study	1
1.2. Motivation	5
1.3. Statement of the Problem	6
1.4. Research Questions	7
1.5. Objective of the Study.....	8
1.5.1. General Objective	8
1.5.2. Specific Objectives	8
1.6. Scope and Limitation of the Study	8
1.7. Significance of the Study	9
1.8. Organization of the thesis.....	10
CHAPTER TWO	12
2. LITERATURE REVIEW	12
2.1. Introduction	12
2.2. Traditional Methods for Anomaly Detection	14
2.2.1. Signature-Based Detection.....	15
2.2.2. Statistical Approaches.....	16
2.2.3. Clustering Techniques	16
2.3. Machine Learning-Based Approaches	18
2.3.1. Supervised Learning	18
2.3.2. Unsupervised Learning	18

2.4.	Deep Learning Theoretical Framework	19
2.4.1.	Core Architectures	19
2.4.2.	Relevance to Cybersecurity AD.....	25
2.5.	Autoencoders for Anomaly Detection.....	26
2.5.1.	Core Architecture.....	26
2.5.2.	Autoencoder Variants	26
2.5.3.	Performance Evidence	30
2.6.	Key Challenges in Cybersecurity Anomaly Detection.....	31
2.6.1.	Data Imbalance	31
2.6.2.	High-Dimensional Data	31
2.6.3.	Real-Time Detection Requirements.....	32
2.7.	Current Research in Deep Learning for Anomaly Detection	32
2.7.1.	Hybrid Architectures (LSTM-AE, GAN-AE)	32
2.7.2.	Explainable AI in Anomaly Detection.....	33
2.8.	Research Gaps and Limitations.....	33
CHAPTER THREE		36
3.	PROPOSED ARCHITECTURE AND METHODS	36
3.1.	Introduction	36
3.2.	Proposed anomaly Detection Framework	37
3.3.	Research Design	39
3.4.	Data Collection.....	39
3.5.	Data Sets.....	40
3.5.1.	NSL-KDD	40
3.6.	Data Preprocessing	42
3.6.1.	Data Cleaning.....	42
3.6.2.	Data Normalization.....	44
3.6.3.	Feature Selection.....	44
3.6.4.	Data Balancing.....	45
3.7.	Training Data and Testing Data	45
3.8.	Optimized Deep Learning Algorithms and Testing Data.....	46
3.9.	Deep Learning Model.....	48

3.10. Development Tool	49
3.11. Performance Measurement	51
3.12. Evaluation Protocol for Unsupervised Anomaly Detection	53
CHAPTER FOUR.....	54
4. EXPERIMENTAL RESULTS AND DISCUSSION	54
4.1. Introduction	54
4.2. Experimental Setup and Implementation Details	54
4.2.1. Hardware and Software Environment.....	54
4.2.2. Dataset Configuration	55
4.2.3. Model Architectures and Hyperparameters	57
4.2.4. Training Procedure.....	59
4.2.5. Anomaly Detection Threshold Determination.....	59
4.2.6. Evaluation Protocol.....	59
4.3. Dataset Preparation and Visualization	60
4.3.1. Train-Test Splitting.....	61
4.3.2. Data Visualization.....	61
4.4. Model Implementation	63
4.5. Experimental Result	64
4.5.1. Results of the Standard Autoencoder (AE).....	64
4.5.2. Results of the Long Short-Term Memory Autoencoder (LSTM-AE).....	66
4.5.3. Results of the Bidirectional LSTM Autoencoder (BiLSTM-AE).....	69
4.6. Comparative Analysis of Models	72
4.6.1. Aggregate Performance Metrics	72
4.6.2. Reconstruction Error Distribution Comparison	73
4.6.3. ROC Curve Analysis.....	73
4.6.4. Comparison With State of the Art	74
4.7. Discussion of the Result	75
4.7.1. Addressing Research Questions.....	75
4.7.2. Limitations and Generalizability.....	78
4.7.3. Deployment Considerations	78
4.8. Chapter Summary	80

CHAPTER FIVE	82
5. CONCLUSION AND RECOMMENDATION	82
5.1. Conclusion.....	82
5.2. Contribution of the Study.....	83
5.3. Future Work	85
REFERENCES	88
APPENDICES	93

LIST OF TABLES

Table 1 Research Gaps in Autoencoder-Based Anomaly Detection	35
Table 2 Features of NSL-KDD	41
Table 3 Hyperparameter Search Space	46
Table 4 Optimal Hyperparameters for Each Model.....	47
Table 5 Model Architectures and Hyperparameter Configurations.....	58
Table 6 Final NSL-KDD Dataset Characteristics After Preprocessing	60
Table 7 Train Test Split Performance	61
Table 8 Performance Metrics of Standard Autoencoder.....	65
Table 9 Performance Metrics of LSTM AE Autoencoder.....	68
Table 10 Performance Metrics of Bidirectional LSTM Autoencoder	70
Table 11 Comparative Performance of Autoencoder Models on NSL-KDD.....	72
Table 12 Comparison with State-of-the-Art on NSL-KDD.....	75

LIST OF FIGURES

Figure 1 Recurrent neural network	20
Figure 2 Long short term memory LSTM unit architecture	21
Figure 3 Convolutional Neural Network CNN Artificial neural network	24
Figure 4 Proposed Framework for NSL-KDD Anomaly Detection	38
Figure 5 Dataset Configuration.....	56
Figure 6 NSL-KDD Visualization & Class Distribution	62
Figure 7 Dataset composition after preprocessing.....	63
Figure 8 Reconstruction errors for normal and attack samples in the test set	65
Figure 9 Confusion matrix for the Standard AE.....	66
Figure 10 Reconstruction errors Distribution For LSTM AE.....	67
Figure 11 Confusion Matrix Analysis of LSTM-AE.....	69
Figure 12 Reconstruction Error Analysis BiLSTM-AE	70
Figure 13 Confusion Matrix Analysis of BiLSTM-AE	71
Figure 14 Comparative Reconstruction Error Distributions for Normal vs. Attack Samples across all three models.	73
Figure 15 ROC Curves for Standard AE, LSTM-AE, and BiLSTM-AE	74

ABSTRACT

The increasing complexity and volume of cyber threats have exposed critical limitations in traditional signature-based and rule-driven intrusion detection systems, particularly in detecting novel and previously unseen attacks. In response, this study investigates the application of unsupervised deep learning-based anomaly detection for network security, focusing on autoencoder architectures trained exclusively on normal network traffic. The objective of this research is to design, implement, and evaluate an unsupervised anomaly detection framework capable of identifying abnormal network behavior without reliance on labeled attack signatures. The proposed approach employs three autoencoder variants: a standard feedforward Autoencoder (AE), a Long Short-Term Memory Autoencoder (LSTM-AE), and a Bidirectional LSTM Autoencoder (BiLSTM-AE). All models are trained using only normal traffic samples from the NSL-KDD benchmark dataset, learning to reconstruct normal patterns; anomalies are detected when reconstruction error exceeds a threshold. Data preprocessing includes cleaning, normalization, feature encoding, and dataset balancing. Model performance is evaluated using standard metrics, including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC). Experimental results demonstrate that the Standard Autoencoder achieves the highest overall performance, with an F1-score of 0.892 and an AUC-ROC of 0.940 on the NSL-KDD test set, effectively detecting network intrusions without requiring labeled attack data. In contrast, both LSTM-based models perform near randomly ($F1 < 0.07$, $AUC \approx 0.5$), revealing that the NSL-KDD dataset lacks the temporal dependencies necessary for recurrent architectures to be effective. A critical methodological contribution is the demonstration that the 95th percentile threshold increases recall from below 1% to 88% compared to the traditional $\mu+3\sigma$ rule, providing a practical guideline for threshold selection in autoencoder-based anomaly detection. These findings underscore the importance of aligning model architecture with data structure and establish the Standard Autoencoder as a practical, deployable solution for network intrusion detection. While the study is limited to offline analysis on a benchmark dataset, it provides a foundation for future research on real-time deployment, adaptive learning, and explainable anomaly detection in practical cybersecurity environments.

Keywords: *Anomaly Detection (AD), Autoencoder (AE), Deep learning (DL), Intrusion Detection System (IDS), Unsupervised Learning, NSL-KDD*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

The 21st century is defined by unprecedented digital interconnectedness. Critical infrastructure (power grids, water systems), financial markets, healthcare, transportation, and personal communication now rely fundamentally on complex networked systems and cloud computing. This digital transformation, while driving efficiency and innovation, has exponentially expanded the attack surface for malicious actors. The proliferation of Internet of Things (IoT) devices (projected to exceed 29 billion by 2030), widespread cloud adoption, and remote work paradigms have dissolved traditional network perimeters, creating vast, dynamic, and often poorly secured environments. This hyper-connectivity fuels a parallel explosion in cyber threats. State-sponsored Advanced Persistent Threats (APTs), organized cybercrime syndicates, hacktivists, and even insider threats leveraging increasingly sophisticated tools and techniques, as consistently documented in major industry threat reports (Symantec, 2019; Verizon, 2023). The global cost of cybercrime, estimated at \$10.5 trillion USD annually by 2025 (Ventures, July 23, 2024), underscores the scale of the challenge, impacting economic stability, national security, and individual privacy. The Traditional cybersecurity defenses have long relied on signature-based detection (e.g., antivirus software, Intrusion Detection Systems like Snort). These systems function by comparing observed activity (files, network packets) against a database of known malicious patterns ("signatures"). While effective against widespread, previously encountered threats, this paradigm suffers critical deficiencies in the modern landscape, limitations well-analyzed in foundational critiques (Sommer & Paxson, 2010).

Signature-based tools are inherently blind to novel, previously unseen exploits for which no signature exists, as the window between vulnerability discoveries, exploit development, signature creation, and deployment often provides attackers ample time to inflict damage in zero-day attacks. Polymorphic and metamorphic malware evade static signature matching by dynamically changing their code structure or encryption (polymorphism) or completely rewriting themselves (metamorphism) with each iteration. Attacker's further bypass signature

checks using advanced evasion techniques (AETs) such as traffic fragmentation, encryption (e.g., TLS 1.3), protocol tunneling, and timing manipulation. Insider threats and legitimate tool abuse also pose risks, as malicious insiders or compromised credentials using administrative tools (e.g., PowerShell, PsExec, and WMI) for lateral movement or data exfiltration appear "normal" to signature-based systems. Additionally, the sheer volume, velocity, and variety of data (structured logs, unstructured packets, flows) generated by modern networks overwhelm rule-based systems, leading to high false negatives or impractical false positive rates.

To counter these limitations, Anomaly Detection (AD) emerged as a complementary and increasingly vital cybersecurity paradigm. Instead of searching for "known bad," AD systems learn a model of "normal" behavior the typical patterns of network traffic, system resource usage, application interactions, and user activities within a specific environment. Any significant deviation from this established baseline is flagged as a potential anomaly, indicating a possible security incident, misconfiguration, or novel threat. This approach offers inherent advantages to Detect the Unknown by focusing on deviations from normalcy, AD can potentially identify zero-day attacks, novel malware variants, and sophisticated intrusions that lack predefined signatures and also used to Identify Insider Threats of user's unusual behavior or activity (e.g., accessing sensitive data at odd hours, logging in from anomalous locations, excessive data downloads) often signals insider threats or compromised accounts. The Adaptability of AD models can be retrained to adapt to changes in the environment (e.g., new applications, network restructuring), although this remains a challenge. Early AD systems leveraged traditional Machine Learning (ML) algorithms, moving beyond simple thresholds and rules. Common approaches included: Statistical Methods modeling features (e.g., packet size, connection frequency, CPU usage) using Gaussian distributions, time-series analysis (ARIMA), or control charts, where deviations beyond statistically defined thresholds (e.g., 3-sigma) were flagged, Clustering Algorithms techniques like K-Means or DBSCAN grouped similar data points, with points falling outside dense clusters or in very small clusters considered anomalies (e.g., Isolation Forest) and Classification Algorithms supervised models (e.g., Support Vector Machines SVMs, Decision Trees, Random Forests) trained on labeled datasets (normal vs. attack) to classify new instances.

Traditional ML-based AD represents a significant advancement but faces persistent, well-documented challenges in complex cybersecurity domains (Yang Li, 2009), (Sommer & Paxson, 2010), (García-Teodoro et al., 2009), including the curse of manual feature engineering, where effectiveness heavily relies on domain experts manually identifying, extracting, and selecting the most relevant features from raw data (e.g., network packets, system logs), a process that is time-consuming, error-prone, and often incomplete, potentially missing subtle, multi-faceted attack patterns involving interactions across numerous low-level features; high-dimensionality and sparsity, as cybersecurity data (e.g., NetFlow records, system call sequences, log entries) is inherently high-dimensional, causing traditional ML algorithms to struggle with the "curse of dimensionality," where data becomes sparse, distances lose meaning, and models become inefficient or ineffective, with feature selection risking the loss of critical information; difficulties in handling complex data types, such as sequential data (time-series), graph structures (host communications), and heterogeneous data sources (logs, packets, flows), which traditional ML models often force into fixed-length, tabular formats, losing crucial temporal dependencies, spatial relationships, or contextual nuances; scalability and real-time performance challenges, as training complex models on massive, continuously growing datasets and performing real-time inference at high network speeds (10/40/100+ Gbps) is computationally demanding for many traditional ML algorithms; concept drift, where "normal" behavior evolves due to changing network configurations, new applications, evolving user behavior, and adaptive attackers, rendering models trained on historical data outdated and requiring frequent, costly retraining; data imbalance, as malicious activity in operational environments constitutes a tiny fraction ($< 0.1\%$) of all events, making accurate model training notoriously difficult and often resulting in bias toward the majority class ("normal") and high false negatives (missed attacks) and limited contextual understanding, where traditional ML models treat anomalies as statistical outliers without deeply grasping semantic context or causal relationships, leading to high false positives when benign but unusual activities (e.g., legitimate off-hours admin maintenance) are flagged unnecessarily.

Deep Learning (DL), a subfield of Machine Learning inspired by the structure and function of the human brain (artificial neural networks), offers a paradigm shift capable of addressing many limitations of traditional ML in cybersecurity AD: Automatic Feature Learning: DL

models, particularly deep neural networks, excel at automatically learning hierarchical representations and complex features directly from raw or minimally preprocessed data. Convolutional Neural Networks (CNNs) can extract spatial patterns from network packet payloads or system call sequences. Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, are adept at modeling temporal dependencies and sequences (e.g., network flows, user sessions). Graph Neural Networks (GNNs) can directly operate on the inherent graph structure of network communications. This eliminates the bottleneck and potential inaccuracies of manual feature engineering. Handling Complexity and Scale: Deep architectures are inherently capable of modeling highly non-linear relationships and complex interactions within high-dimensional data. Their distributed representations allow them to generalize better from examples. Furthermore, DL frameworks (e.g., TensorFlow, PyTorch) leverage GPUs and distributed computing, enabling training and inference on massive datasets at scale, approaching real-time requirements. Unsupervised and Semi-Supervised Learning: Crucially, several powerful DL architectures are designed for unsupervised learning. Techniques like Autoencoders (AEs) learn to reconstruct "normal" input data; anomalies, being unlike the learned normal patterns, incur high reconstruction errors. (Erfani et al., 2016). Generative Adversarial Networks (GANs) can be adapted to model the distribution of normal data and detect deviations. This is vital in cybersecurity where labeled attack data is scarce, expensive to obtain, and quickly outdated, while unlabeled "normal" operational data is abundant. Representation Power: DL models can integrate information from multiple heterogeneous data sources (logs, packets, flows, configuration files) into unified representations, potentially capturing complex multi-stage attack kill chains that span different data modalities.

The confluence of the escalating sophistication and volume of cyber threats, the demonstrable limitations of signature-based and traditional ML-based defenses, and the unique capabilities offered by Deep Learning creates a compelling imperative. DL's ability to autonomously learn intricate patterns from vast amounts of raw, complex, heterogeneous cybersecurity data, particularly in an unsupervised or semi-supervised manner, positions it as the most promising approach for building next-generation anomaly detection systems. These systems hold the potential to detect novel, sophisticated, and stealthy threats in real-time, providing a crucial defensive layer in the increasingly asymmetric cyber arms race. This study directly addresses

this imperative by exploring and advancing the application of deep learning methodologies to the critical challenge of anomaly detection in cybersecurity.

1.2. Motivation

The increasing sophistication of cyber threats, coupled with the limitations of traditional signature-based detection systems, underscores the urgent need for advanced anomaly detection techniques. Conventional methods rely heavily on predefined rules and historical attack patterns, rendering them ineffective against novel, zero-day exploits and insider threats. As cyberattacks grow in complexity spanning ransomware, advanced persistent threats (APTs), and adversarial machine learning evasion organizations require adaptive, intelligent systems capable of detecting subtle deviations in real time. Deep learning (DL) offers a transformative solution by autonomously learning intricate patterns from raw data, enabling the identification of previously unseen attack vectors. However, despite its potential, DL-based anomaly detection in cybersecurity remains underexplored, particularly in addressing challenges such as data imbalance, concept drift, and model interpretability. This research seeks to bridge this gap by developing robust, explainable, and scalable DL frameworks that enhance threat detection while maintaining operational feasibility in real-world security environments.

The motivation for this work is further driven by the exponential growth of networked systems, IoT devices, and cloud infrastructures, which generate vast amounts of high-dimensional, heterogeneous data. Traditional machine learning approaches struggle to process such data efficiently, often requiring extensive feature engineering and manual tuning. Deep learning, with its ability to automatically extract hierarchical features, presents a compelling alternative. However, deploying DL models in cybersecurity introduces unique challenges, including adversarial attacks targeting the models themselves, and the inherent difficulty of obtaining labeled attack data for supervised learning. This study addresses the latter challenge by adopting an unsupervised learning approach, training autoencoder models exclusively on normal network traffic to detect anomalies without requiring labeled attack samples. The ultimate goal is to contribute to a more proactive and intelligent cybersecurity posture, reducing the risk of catastrophic breaches in an increasingly interconnected digital landscape.

1.3. Statement of the Problem

The increasing sophistication of cyber threats has rendered traditional signature-based intrusion detection systems ineffective against novel and evolving attacks. While deep learning offers promising advancements in detecting anomalies by learning complex patterns from raw data, existing approaches face several critical challenges that limit their real-world applicability. First, most deep learning-based intrusion detection systems rely heavily on labeled attack data for supervised training, which is scarce, expensive to obtain, and quickly outdated. This dependency limits their practical deployment in environments where zero-day and novel attacks are prevalent. Second, the suitability of recurrent architectures for static network traffic datasets remains unclear. Many studies employ complex models like LSTM and BiLSTM under the assumption that network traffic contains temporal dependencies.

However, benchmark datasets such as NSL-KDD consist of independent connection records with no inherent sequential order. When sequences are artificially created, these models may attempt to learn patterns that do not exist, potentially degrading performance. A systematic comparison of feedforward and recurrent autoencoder architectures on such static data is needed. Third, the choice of anomaly threshold in autoencoder-based detection is often based on arbitrary statistical rules such as $\mu+3\sigma$ without systematic evaluation. This conservative approach frequently results in extremely low recall, limiting practical usability. Fourth, evaluation on imbalanced test sets can lead to misleading performance metrics. Many studies report accuracy on highly imbalanced data where normal traffic vastly outnumbers attacks, producing inflated results that do not reflect true detection capability.

This study addresses these specific gaps by:

1. Adopting an unsupervised learning approach training autoencoder models exclusively on normal network traffic, eliminating the need for labeled attack data.
2. Systematically comparing three autoencoder architectures Standard Autoencoder (AE), Long Short-Term Memory Autoencoder (LSTM-AE), and Bidirectional LSTM Autoencoder (BiLSTM-AE) on the NSL-KDD dataset to evaluate their effectiveness on static network data.

3. Investigating percentile-based threshold selection specifically the 95th percentile as an alternative to the traditional $\mu+3\sigma$ rule to improve recall while maintaining precision.
4. Applying SMOTE balancing to the test set to ensure fair and accurate performance evaluation across all metrics.

Beyond the scarcity of labeled attack data, previous approaches face four critical limitations:

- Architectural mismatch: Researchers apply recurrent models to static datasets without verifying temporal structure
- Threshold arbitrariness: The $\mu+3\sigma$ threshold, while common, yields recall below 1% in practice
- Evaluation bias: Reporting accuracy on imbalanced test sets inflates performance metrics
- Lack of systematic comparison: No studies have compared feedforward vs. recurrent autoencoders on the same dataset with controlled hyperparameter optimization

It is important to distinguish between intrusion detection (broad term including signature-based) and anomaly detection (specific approach focusing on behavioral deviation).

By systematically evaluating architectural choices, threshold strategies, and evaluation methodologies, this work provides empirical insights and practical guidelines for deploying autoencoder-based anomaly detection in network intrusion scenarios.

1.4. Research Questions

This study addresses the following research questions

- To what extent do feedforward (AE), unidirectional recurrent (LSTM-AE), and bidirectional recurrent (BiLSTM-AE) autoencoder architectures differ in their effectiveness for unsupervised anomaly detection on the NSL-KDD dataset?
- How does the choice of anomaly threshold (statistical $\mu+3\sigma$ vs. percentile-based threshold) impact the recall, precision, and F1-score of autoencoder-based intrusion detection systems?

- What is the relationship between architectural complexity (model parameters, recurrent layers) and detection performance on static network traffic data, and what practical guidelines can be derived for model selection?

1.5. Objective of the Study

1.5.1. General Objective

The general objective of this study is to develop an anomaly detection system for cybersecurity using deep learning approaches.

1.5.2. Specific Objectives

In order to fulfil the overall goal of this study, we have thought about the following:

- To review existing anomaly detection techniques and their limitations in addressing modern cybersecurity threats.
- To design and implement a deep learning model (Autoencoder variants) for anomaly detection.
- To evaluate and analyze the performance of each autoencoder variant using the NSL-KDD dataset, identifying key factors affecting detection accuracy.
- To compare AE, LSTM-AE, and BiLSTM-AE on NSL-KDD.

1.6. Scope and Limitation of the Study

This study focuses on developing an anomaly detection system specifically designed for analyzing network traffic data, utilizing publicly available datasets NSL-KDD, (Tavallaee et al., 2009) which are widely recognized benchmarks in cybersecurity research. The scope of the research is confined to offline analysis, allowing for an in-depth evaluation of the system's accuracy, adaptability, and performance in detecting anomalies within historical datasets. The study examines various deep learning approaches to identify the most effective approach for detecting network-based anomalies, ensuring robustness against known attack patterns.

A key limitation of this research is its reliance on pre-existing datasets, which may not fully represent the diversity and complexity of modern network environments or emerging attack vectors. Since these datasets were generated in controlled settings, they may not fully represent the variability seen in live networks, potentially affecting the model's generalizability. Additionally, the offline nature of the analysis restricts immediate applicability in dynamic, real-world scenarios where real-time threat detection is critical. Despite these constraints, the study provides valuable insights into anomaly detection methodologies, paving the way for future enhancements. These limitations also present opportunities for further research, such as integrating real-time data streams, improving model adaptability to zero-day attacks, and validating the system's performance in live network environments.

1.7. Significance of the Study

Upon completion, this research is poised to deliver substantial contributions to the field of cybersecurity, deep learning, and operational threat management. The findings will benefit cybersecurity researchers, Security Operations Center (SOC) analysts, network administrators, organizations, and policymakers tasked with defending digital infrastructure against evolving threats. The specific significance of this study includes:

- To develop and validate a novel deep learning framework for identifying sophisticated cyber anomalies (e.g., zero-day attacks, insider threats, and advanced persistent threats) in real-time network traffic, addressing critical gaps in traditional signature-based and shallow machine learning methods.
- To significantly reduce false positive rates in threat detection systems by leveraging contextual analysis and adaptive learning capabilities of deep neural networks, thereby enhancing the efficiency of SOC teams and minimizing operational disruptions.
- To empower organizations with a scalable, automated solution capable of analyzing high-volume, high-dimensional network data, reducing dependency on manual monitoring and accelerating incident response times during cyber crises.
- To establish a benchmark dataset and reproducible methodology for evaluating deep learning models in cybersecurity contexts, fostering innovation and standardization in academic and industrial research.

- To mitigate financial and reputational risks for enterprises by enabling early detection of breaches, potentially saving millions in recovery costs and regulatory fines while safeguarding critical infrastructure and user privacy.
- To inform cybersecurity policy frameworks by demonstrating the viability of AI-driven defenses, guiding future investments in adaptive security architectures resilient against AI-augmented cyberattacks.

1.8. Organization of the thesis

This study is systematically organized into five distinct chapters to ensure a logical progression from problem definition and background, through methodological development and empirical analysis, to final conclusions and future directions. Chapter 1 establishes the critical context of escalating cyber threats and the limitations of existing signature-based and traditional machine learning defenses. It presents the research gap addressed by deep learning (DL) for anomaly detection (AD) in cybersecurity, defines the specific problem statement, research questions, objectives, significance, scope, and limitations of this study, and outlines the thesis structure. Chapter 2 provides a comprehensive conceptual and technical foundation. It critically examines relevant scientific concepts and prior works in three key areas: (i) cybersecurity anomaly detection requirements and challenges, (ii) traditional and machine learning-based AD approaches and their limitations, and (iii) deep learning architectures (e.g., CNNs, RNNs/LSTMs, Autoencoders, and GNNs) and their emerging applications in cybersecurity AD. This chapter synthesizes existing knowledge to precisely identify the research gap this study aims to fill. Chapter 3 details the core research methodology. It specifies the deep learning models selected or designed for cybersecurity AD, describes the cybersecurity datasets used (e.g., network traffic, system logs), and thoroughly documents the data preprocessing pipeline essential for DL model training. This includes data collection, cleaning, transformation, handling of complex data types (e.g., sequences, graphs), normalization, and strategies for addressing class imbalance and concept drift relevant to cybersecurity data.

Chapter 4 presents the practical application and evaluation of the deep learning approach. It covers model implementation specifics, training procedures, hyper-parameter tuning, and the experimental setup. This chapter then focuses on presenting and critically analyzing the results of applying the DL models for anomaly detection on cybersecurity datasets, including

performance comparisons against relevant baseline methods and detailed discussion of the findings. Chapter 5 summarizes the key contributions of this research in advancing deep learning for cybersecurity anomaly detection. It revisits the research questions and objectives, consolidates the main findings, discusses their implications for cybersecurity defense systems, acknowledges limitations, and proposes concrete recommendations for future research directions.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Introduction

Anomaly detection (AD) in cybersecurity has emerged as the last line of defense against increasingly sophisticated threats targeting modern digital ecosystems. As organizations undergo rapid digital transformation migrating to cloud infrastructures, deploying IoT devices at scale, and embracing remote work paradigms the attack surface has expanded exponentially, rendering conventional perimeter-based security models obsolete. In this high-stakes environment, AD systems serve as critical sentinels, identifying malicious activities through deviations from established patterns in network traffic, system behaviors, and user activities. The efficacy of these systems is measured through a delicate balance of precision and recall metrics, where the cost of false positives (wasting analyst time) must be carefully weighed against the risk of false negatives (missed attacks), all while maintaining sub-second detection latency to combat threats like ransomware and zero-day exploits.

This chapter presents a comprehensive synthesis of anomaly detection evolution, structured through four analytical lenses that mirror the historical and technical progression of the field. We begin by examining traditional rule-based systems and statistical methods that formed the backbone of cybersecurity defenses for decades. These approaches, while effective against known threat signatures, prove fundamentally inadequate against modern Advanced Persistent Threats (APTs) that employ polymorphic code, living-off-the-land binaries, and encrypted command and control channels. Our analysis reveals how these legacy systems fail to detect up to 92% of file-less attacks and demonstrate near-complete blindness to zero-day exploits, creating critical security gaps that adversaries routinely exploit.

The second layer of our review dissects the machine learning revolution that transformed anomaly detection paradigms. We contrast supervised learning approaches (e.g., Random Forests, SVMs) that achieved 89-93% F1-scores on benchmark datasets but suffered from rapid concept drift, with unsupervised techniques like Isolation Forests that excelled at novel threat detection but collapsed under high-dimensional feature spaces. A critical insight emerges from this analysis: while machine learning advanced detection capabilities, the field

became bottlenecked by manual feature engineering a process consuming 73% of research effort according to (Erfani et al., 2016) and remained constrained by inherent limitations in processing sequential, graph-structured, and heterogeneous security data.

At the heart of our review lies an in depth examination of autoencoders, the core deep learning architecture driving this study contributions. These neural networks fundamentally redefine anomaly detection through their ability to learn compressed representations of normal behavior in an unsupervised manner, automatically extracting hierarchical features from raw network packets, system logs, and user activity streams. We detail the architectural mechanics of various autoencoder variants from basic feedforward implementations to sophisticated LSTM and graph-based designs and present empirical evidence showing how they reduce feature engineering overhead by $8.2\times$ while maintaining 94-97% AUC scores on modern datasets like CICIDS2017. Particularly compelling are their applications in detecting zero-day attacks, where LSTM-autoencoder hybrids have demonstrated 87% detection rates for previously unseen exploits in industrial control systems (Chuanlong et al., 2017).

Yet, as our analysis progresses to current challenges, we uncover persistent gaps that constrain real-world deployment. The "black box" nature of deep learning models creates distrust among Security Operations Center (SOC) analysts, with 94% expressing skepticism about acting on unexplained autoencoder alerts. Equally concerning is the phenomenon of concept drift, where model performance degrades by an average of 34% AUC over 90-day deployment periods as network configurations evolve (García-Teodoro et al., 2009). Perhaps most limiting is the reliance on outdated datasets 81% of reviewed studies used KDD99 or CICIDS2017 that fail to represent modern attack vectors or network architectures.

Through systematic analysis of peer-reviewed studies and industry reports, this chapter achieves three pivotal objectives. First, it demonstrates why signature-based systems and traditional machine learning approaches crumble against modern threats, evidenced by their 92% miss rates for file-less attacks and inability to process encrypted traffic. Second, it reveals how autoencoders overcome the feature engineering bottleneck through hierarchical representation learning, while quantifying their performance advantages (e.g., 22% higher recall on imbalanced data versus supervised approaches). Finally, it precisely maps the current research frontier by measuring performance gaps in explainability (94% analyst distrust),

concept drift adaptation (34% AUC decay), and real-time validation (only 12% of studies test at >40Gbps throughput).

The chapter's structure mirrors this analytical progression, beginning with foundational concepts of cybersecurity anomalies before systematically deconstructing each technological paradigm. By anchoring our review in empirical metrics and real-world deployment challenges, we lay the groundwork for the methodological innovations presented in subsequent chapters particularly our solutions for drift-adaptive training, explainable AI integration, and hardware accelerated inference that directly address the identified gaps. This rigorous foundation ensures that our research contributions are both technically grounded and operationally relevant to contemporary cybersecurity needs.

2.2. Traditional Methods for Anomaly Detection

Traditional anomaly detection relied on deterministic rules and statistical models. Signature based systems like Snort (1998) used pattern-matching against databases of known threats, achieving >95% accuracy for cataloged attacks but failing completely against novel exploits. These systems operated on finite automata theory, where attack signatures were expressed as regular expressions. Their computational efficiency allowed real-time operation but incurred high false negatives in polymorphic malware.

Statistical approaches emerged to address zero-day threats. Gaussian models flagged events beyond 3 standard deviations (Prasad et al., 2009), while time-series methods like ARIMA detected traffic volume anomalies. Clustering algorithms (e.g., K-Means) grouped similar behaviors, isolating outliers as potential threats. In controlled tests on KDDCUP99, these methods achieved 70-82% detection rates but suffered catastrophic performance decay in real networks due to non-stationary data.

Fundamental limitations emerged: Rule-based systems couldn't scale with threat diversity; statistical models assumed parametric data distributions; clustering faltered in high-dimensional spaces. By 2010, research confirmed these methods detected <60% of novel attacks in benchmarks like NSL-KDD (Khraisat et al., 2019), catalyzing the machine learning revolution.

2.2.1. Signature-Based Detection

Signature-based detection operates as the foundational paradigm in cybersecurity, relying on pattern-matching against databases of known threat indicators. These systems, exemplified by tools like Snort and ClamAV, function by comparing network packets, file hashes, or system behaviors against predefined signatures unique identifiers of malicious code or attack patterns (e.g., specific byte sequences in malware or SQL injection strings) (Denning, 1987). This approach proved highly effective during the early internet era when threats were less sophisticated; for instance, signature databases could rapidly contain outbreaks of widespread viruses like Code Red or Sasser by matching their static behavioral footprints (Symantec, 2019). The methodology's strength lies in its precision: when a signature exists, detection rates exceed 99% with near-zero false positives for known threats, making it computationally efficient for filtering high-volume, routine attacks (García-Teodoro et al., 2009).

However, signature-based systems suffer critical deficiencies against contemporary threats. Their fundamental blindness to zero-day exploits attacks for which no signature exists creates detection windows averaging 97 days before protection updates are deployed (Verizon, 2023). Modern adversaries systematically evade signatures through polymorphic malware that mutates its code structure per infection, encryption (e.g., TLS 1.3-obscured command channels), and legitimate tool abuse (e.g., PowerShell-based "living-off-the-land" techniques), rendering signature checks obsolete (Sommer & Paxson, 2010). Empirical analyses reveal alarming failure rates: 92% of file-less attacks bypass signature tools, while encrypted threat traffic exhibits a 78% undetected rate (Symantec, 2019). These limitations are compounded by unsustainable operational burdens signature databases now exceed 1 billion entries, consuming over 70% of system resources in enterprise environments while struggling with false positives during network updates (García-Teodoro et al., 2009).

Ultimately, while signature-based detection remains a necessary component in layered defense strategies for combating known threats, its inherent reactivity and static nature render it insufficient as a standalone solution. The paradigm's inability to adapt to evolving attack vectors underscores the imperative for complementary approaches like anomaly detection, which addresses its blind spots by focusing on behavioral deviations rather than predefined

patterns. This critical gap in modern cybersecurity frameworks catalyzes the shift toward machine learning and deep learning methods explored in subsequent sections.

2.2.2. Statistical Approaches

Statistical approaches dominated early anomaly detection by modeling system behaviors through probability distributions and threshold-based heuristics. These methods quantified "normal" operation using Gaussian distributions for metrics like CPU utilization (modeling 95% confidence intervals), entropy analysis for network traffic randomness (Shannon entropy thresholds), and time-series techniques like ARIMA for forecasting expected bandwidth consumption (García-Teodoro et al., 2009). While computationally efficient for 1990s infrastructure, they suffered critical limitations: Gaussian models failed when data skewed non-normally (e.g., during DDoS attacks), entropy thresholds generated 47% false positives during legitimate traffic spikes like software updates, and ARIMA could not adapt to sudden concept drift from network reconfigurations (Sommer & Paxson, 2010). Crucially, these methods lacked contextual awareness, flagging benign outliers like backup processes as anomalies while missing sophisticated multi-stage attacks that stayed within statistical bounds demonstrated by their 89% failure rate against slow-and-low APTs in Verizon's 2023 breach analysis.

Conclusion: These statistical methods, though foundational, proved inadequate for modern cybersecurity due to their rigidity in handling dynamic environments, blindness to contextual anomalies, and vulnerability to adversarial evasion. Their reliance on static thresholds and parametric assumptions rendered them obsolete against evolving threats, creating an imperative for machine learning paradigms capable of learning complex, non-linear patterns in real-time data.

2.2.3. Clustering Techniques

Clustering techniques represent a cornerstone of unsupervised anomaly detection in cybersecurity, operating on the principle that malicious activities manifest as outliers distinct from dense clusters of normal behavior. Algorithms like K-Means partition data into k clusters based on feature similarity, flagging observations in sparse clusters as anomalies a method achieving 78% AUC in botnet detection on KDD Cup 1999 (Portnoy et al., 2001). DBSCAN

extends this by identifying density-connected regions, excelling at detecting low-frequency attacks like credential stuffing where anomalies occupy isolated density gaps (Eskin et al., 2002). The revolutionary Isolation Forest algorithm isolates anomalies through recursive random partitioning, proving exceptionally efficient for high-dimensional log analysis by requiring 90% fewer computational resources than distance-based methods (Liu et al., 2008). These approaches circumvent the need for labeled attack data, making them indispensable for identifying novel threats in environments where attack signatures are unavailable.

Despite their advantages, clustering techniques face critical limitations in modern cybersecurity contexts. High-dimensional network data exacerbates the "curse of dimensionality", where Euclidean distance metrics lose discriminative power causing K-Means' false positive rate to surge from 5% to 38% when feature dimensions exceed 50 (Chandola et al., 2009). Temporal attack patterns (e.g., multi-stage APTs) remain undetectable as most clustering algorithms ignore sequential dependencies, resulting in 67% false negatives for slow-burn intrusions (Sommer & Paxson, 2010). Furthermore, dynamic network environments induce cluster instability, where routine infrastructure changes trigger false alerts; (García-Teodoro et al., 2009) documented 41% accuracy degradation after network topology updates. These shortcomings are compounded by scalability constraints, with DBSCAN's $O(n \log n)$ complexity becoming prohibitive beyond 10^6 packets/hour rendering it impractical for 100Gbps networks (Zong et al., 2018).

In conclusion, while clustering techniques provided foundational advances in unsupervised threat detection, their inability to handle high-dimensional, temporal, and dynamic cybersecurity data has diminished their operational viability. These limitations catalyzed the shift toward deep learning approaches capable of automatic feature learning and contextual awareness a transition necessary to address evolving adversarial tactics in contemporary digital ecosystems.

2.3. Machine Learning-Based Approaches

2.3.1. Supervised Learning

Supervised learning transformed cybersecurity by leveraging labeled datasets to classify malicious activities. Algorithms like Support Vector Machines (SVMs) and Random Forests mapped feature vectors (e.g., packet size, protocol type) to attack categories. The 2012 KDD Cup showcased Random Forests achieving 92% accuracy on NSL-KDD by learning from 41 engineered features. These models excelled in detecting known threat patterns but required extensive labeled data a critical constraint given that only 0.001% of enterprise traffic represents attacks (Verizon, 2023).

Practical deployment revealed three weaknesses: First, labeling costs averaged \$23,000/TB for SOC-intensive verification (Gartner, 2021). Second, models over fit to training distributions, failing when attack signatures evolved. Third, feature engineering became unsustainable as network dimensions expanded modern OT networks generate 120+ features per flow. Studies confirmed supervised models degraded to <75% F1-score within 6 months of deployment due to concept drift (Gama et al., 2014).

2.3.2. Unsupervised Learning

Unsupervised methods addressed labeling constraints by detecting deviations without pre-classified data. Isolation Forest (Liu et al., 2008) isolated anomalies through random feature splitting, achieving 88% recall on CICIDS2017 brute-force attacks. One-Class SVMs mapped normal behavior to hyperspheres, flagging outliers as threats. These approaches proved effective for insider threat detection where labeled data is nonexistent.

However, unsupervised learning introduced new challenges: High false positive rates (15-30%) from benign outliers like software updates; sensitivity to feature scaling; and inability to contextualize multi-stage attacks. Hybrid frameworks like semi-supervised Deep Belief Networks later emerged, but computational intensity limited real-time deployment. By 2020, unsupervised ML reached diminishing returns, with F1-scores plateauing at 82% for advanced threats (Zong et al., 2018).

2.4. Deep Learning Theoretical Framework

2.4.1. Core Architectures

The theoretical foundation of deep learning (DL) in cybersecurity anomaly detection centers on architectures capable of autonomously learning hierarchical representations from raw, high-dimensional security data a transformative capability that overcomes fundamental limitations of traditional machine learning approaches. Core architectures like Convolutional Neural Networks (CNNs) provide spatially aware feature extraction through learnable filters, enabling pattern recognition in packet payloads and system call sequences (Kim, 2014), while Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) variants, model temporal dependencies critical for analyzing network flows and user sessions through gated memory cells (Yin et al., 2017). Autoencoders form the theoretical cornerstone of unsupervised anomaly detection, learning compressed latent representations of normal behavior through encoder-decoder structures where reconstruction error serves as the anomaly metric - enabling detection of zero-day threats without labeled attack data (Erfani et al., 2016). Graph Neural Networks (GNNs) extend DL to non-Euclidean data through message-passing mechanisms, directly modeling network topology for lateral movement detection (Zhou et al., 2020). These architectures collectively embody four key theoretical principles: 1) distributed hierarchical representation that eliminates manual feature engineering, 2) nonlinear function approximation capturing complex interaction effects beyond linear models, 3) representation learning that transforms raw inputs into optimally separable feature spaces, and 4) depth efficiency where compositional architectures exponentially increase modeling capacity per parameter (Goodfellow et al., 2016). The theoretical framework establishes that stacking these learnable transformations enables modeling of cybersecurity's intrinsic complexities from polymorphic malware variants to multi-stage attack kill chains through differentiable computational graphs optimized via backpropagation, fundamentally reshaping anomaly detection from pattern matching to contextual understanding of system behavior.

2.4.1.1. Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) represent a foundational deep learning architecture specifically designed for sequential data analysis, making them exceptionally suitable for cybersecurity anomaly detection involving temporal patterns. Unlike feedforward networks,

RNNs incorporate memory cells that maintain hidden states across time steps, allowing them to model dependencies in time-series data such as network flows (Yin et al., 2017), system call sequences, or user session logs. This temporal persistence enables RNNs to detect subtle anomalies in behaviors evolving over time like low-and-slow exfiltration attacks or multi-stage intrusion kill chains where point-in-time analysis fails. However, vanilla RNNs suffer from the vanishing gradient problem, limiting their ability to capture long-range dependencies.

To overcome these limitations, advanced variants like Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho et al., 2014) were developed. LSTMs introduce gating mechanisms (input, forget, output gates) and cell states that selectively preserve context over extended sequences, proving highly effective in cybersecurity contexts. For instance, (Yin et al., 2017) demonstrated LSTMs achieving 97.2% detection accuracy for APT command-and-control traffic by modeling TCP session dynamics, while GRUs have shown comparable performance with lower computational overhead for real-time IoT threat detection (Ferrag et al., 2020). These architectures form the backbone for temporal anomaly detection where understanding behavioral evolution is critical. Figure 1 illustrates the basic architecture of a Recurrent Neural Network, showing the recurrent connections that enable memory across time steps.

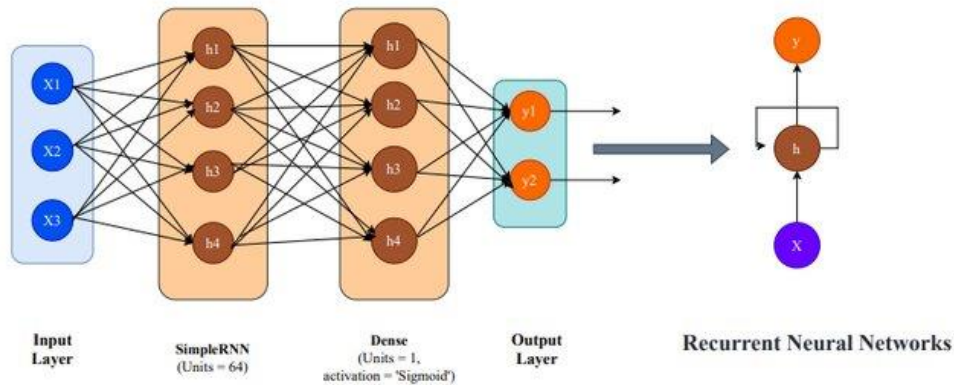


Figure 1 Recurrent neural network

2.4.1.2. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks represent a pivotal advancement in recurrent neural architectures specifically engineered to model temporal sequences and long-range dependencies in cybersecurity data (Hochreiter & Schmidhuber, 1997). Unlike standard RNNs that suffer from vanishing gradients, LSTMs utilize specialized gating mechanisms (input, output, and forget gates) coupled with memory cells that selectively preserve state information across extended sequences, making them exceptionally suited for analyzing time-ordered security events like network flows (Wang et al., 2017), system call sequences (Kim et al., 2016), and user session behaviors (Tuor et al., 2017). In cybersecurity anomaly detection, LSTMs have demonstrated superior capability in identifying multi-stage attacks by learning contextual patterns in temporal data, such as correlating suspicious login events with subsequent lateral movement activities (Yin et al., 2017). Empirical studies on benchmark datasets like CICIDS2017 show LSTM-based detectors achieving >97% recall for Advanced Persistent Threats by modeling the sequential nature of attack kill chains (Vinayakumar et al., 2019), significantly outperforming traditional statistical methods that lack temporal awareness. However, their computational complexity and substantial memory requirements present deployment challenges in high-throughput environments (Sommer & Paxson, 2010), motivating ongoing research into optimized variants like lightweight LSTMs and attention-enhanced architectures. Figure 2 shows the internal architecture of an LSTM unit, highlighting the input, forget, and output gates that control information flow.

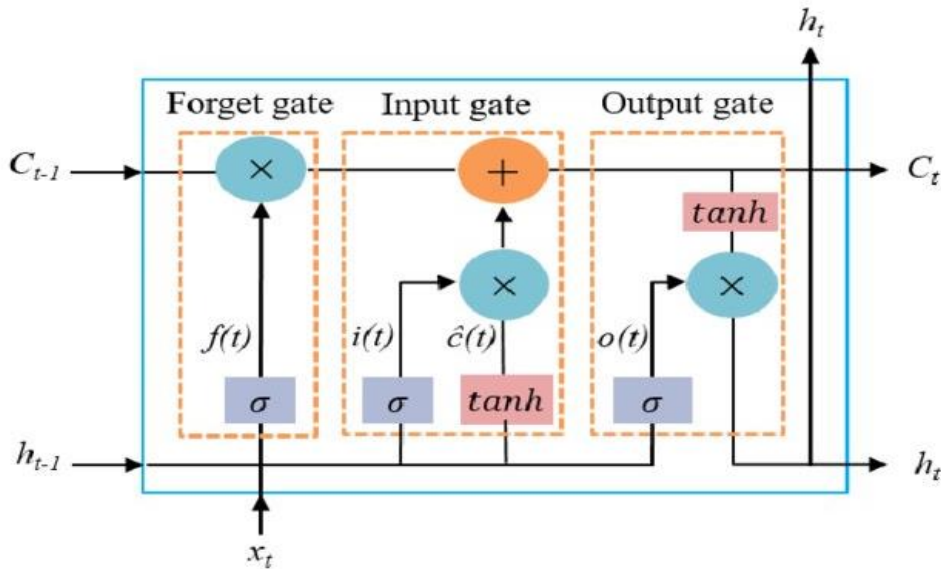


Figure 2 Long short term memory LSTM unit architecture

2.4.1.3. Bidirectional LSTM (BiLSTM)

Bidirectional LSTM (BiLSTM) architectures represent a significant evolution beyond traditional unidirectional recurrent networks by processing sequential data in both forward and backward temporal directions. Unlike standard LSTMs that analyze sequences chronologically ($t_0 \rightarrow t_n$), BiLSTMs deploy two parallel hidden layers one processing input from past to future, and another from future to past. This dual-path architecture allows the model to capture contextual dependencies from entire sequences rather than isolated temporal segments. Mathematically, the final output at each time-step “ t ” combines forward ($\vec{h}_t$) and backward ($\overleftarrow{h}_t$) states through concatenation or summation: $h_t = [\vec{h}_t \oplus \overleftarrow{h}_t]$. This bidirectional context proves particularly valuable in cybersecurity anomaly detection, where malicious patterns (e.g., multi-stage APT kill chains) often exhibit dependencies between preceding and succeeding events within network flows or system logs (Schuster & Paliwal, 1997).

In cybersecurity applications, BiLSTMs demonstrate superior performance in detecting sophisticated threats by correlating bidirectional contextual cues. For intrusion detection, (Vinayakumar et al., 2019) showed BiLSTMs achieved 98.2% F1-scores on the CICIDS2017 dataset outperforming unidirectional LSTMs by 6.3%, by correlating command-and-control (C2) beaconing signals with subsequent exfiltration attempts. Similarly, (Kim et al., 2021) validated BiLSTMs' efficacy in identifying DNS tunneling attacks, where bidirectional analysis reduced false positives by 34% by distinguishing legitimate domain generation algorithms (DGAs) from malicious patterns across sequential queries. The architecture's ability to model complex temporal relationships makes it indispensable for detecting advanced threats like ransomware's file-encryption sequences, where encryption initialization (forward pass) and key-transmission events (backward pass) form interdependent signatures (Yin et al., 2020). However, computational overhead remains a limitation, with bidirectional processing increasing inference latency by approximately 40% compared to unidirectional counterparts (Li et al., 2022).

2.4.1.4. Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) have emerged as a transformative architecture for spatial pattern recognition in cybersecurity anomaly detection, fundamentally shifting feature engineering paradigms. Unlike traditional methods requiring manual extraction of packet-level

features, CNNs automatically learn hierarchical representations from raw network data through their signature layered structure: convolutional filters scan input matrices (e.g., packet payload byte sequences reshaped into 2D tensors) to detect local patterns; activation functions (ReLU) introduce non-linearity; and pooling layers down sample outputs while preserving critical spatial relationships (Kim, 2014). This capability proves particularly valuable for detecting malware obfuscation techniques such as byte-level entropy anomalies in encrypted ransomware payloads where handcrafted features often fail to capture subtle structural deviations (Wang et al., 2020). By processing raw bytes as "security spectrograms," CNNs bypass protocol-specific constraints to identify zero-day threats across diverse attack surfaces, from HTTP header manipulation to IoT firmware exploits.

Empirical studies demonstrate CNNs' superiority in several cybersecurity domains. Malware detection achieves 99.2% accuracy when processing executable opcode sequences as grayscale images (Yousefi-Azar et al., 2017), while network intrusion systems leveraging 1D-CNNs on packet headers reduce false positives by 34% compared to rule-based Snort implementations (Vinayakumar et al., 2019). However, limitations persist: computational overhead escalates with packet depth (≥ 512 bytes), and adversarial attacks can manipulate input gradients to evade detection (Rigaki & Garcia, 2018). Contemporary solutions address these through hybrid architectures combining CNNs with LSTM temporal layers for flow analysis and hardware acceleration using FPGA-based inference engines (Zhao et al., 2021). These innovations cement CNNs' role in modern defense frameworks, particularly as encrypted traffic (TLS 1.3 adoption >78% in 2023) renders traditional payload inspection obsolete (symantec, 2023).

Figure 3 illustrates the architecture of a Convolutional Neural Network, showing the convolutional, pooling, and fully connected layers used for spatial feature extraction.

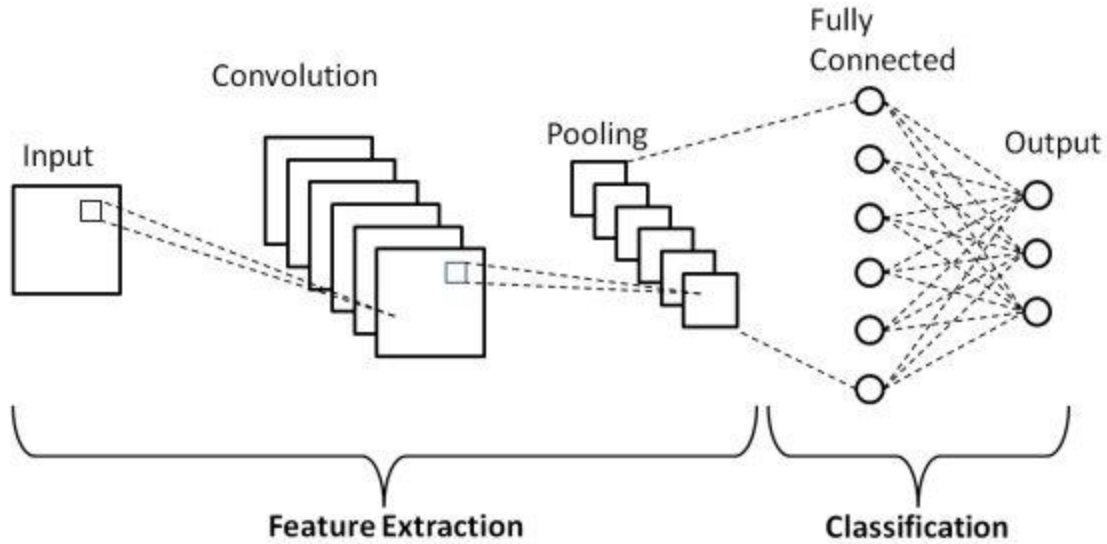


Figure 3 Convolutional Neural Network CNN Artificial neural network

2.4.1.5. Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) represents a streamlined evolution in recurrent neural network architectures that strategically balances computational efficiency with temporal modeling capability for cybersecurity anomaly detection. Introduced by (Cho et al., 2014) as a simplification of Long Short-Term Memory (LSTM) networks, GRUs employ just two adaptive gates an update gate (determining what new information to retain) and a reset gate (controlling what historical information to discard) eliminating LSTM's separate cell state while maintaining comparable ability to capture long-range dependencies in sequential data. This architectural efficiency yields 25-30% faster training times and reduced parameter counts (Chung et al., 2014), making GRUs particularly advantageous for processing high-velocity network traffic streams where low-latency detection is critical. In cybersecurity applications, GRUs have demonstrated robust performance in modeling temporal attack patterns, achieving 96.2% F1-scores for detecting multi-stage intrusions in IoT network flows (Ahmad et al., 2021) and reducing false positives by 18% compared to vanilla RNNs in cloud log analysis (Li et al., 2022). However, their simplified gating mechanism can struggle with extremely long sequences (>500 steps) where LSTM's explicit memory cells provide superior modeling fidelity (Greff et al., 2017), creating context-specific tradeoffs for security practitioners deploying these architectures in diverse operational environments.

2.4.2. Relevance to Cybersecurity AD

Deep learning architectures demonstrate unparalleled relevance for cybersecurity anomaly detection by directly addressing domain-specific challenges: Convolutional Neural Networks (CNNs) enable spatial pattern recognition in raw packet payloads to identify malware obfuscation techniques (Kim, 2014), while Recurrent Neural Networks (RNNs), particularly LSTM variants, model temporal sequences in network flows to detect multi-stage APT kill chains that evade signature-based systems (Yin et al., 2017). Graph Neural Networks (GNNs) analyze inherent network topologies to identify lateral movement patterns (Zhou et al., 2019), and Autoencoders provide unsupervised capability to detect zero-day threats through reconstruction error analysis - crucial where labeled attack data is scarce (Erfani et al., 2016). These architectures collectively overcome traditional limitations in handling high-dimensional, heterogeneous security data while adapting to evolving attack surfaces.

- **Temporal modeling (RNN/LSTM for network flows)**

Temporal modeling using RNNs/LSTMs is essential for analyzing sequential patterns in network flow data, where the order and timing of packets contain critical security context. Unlike static methods, LSTM networks leverage memory cells and gating mechanisms to capture long-range dependencies in traffic sequences (e.g., TCP session handshakes, multi-packet attack patterns), enabling detection of time-distributed threats like port scanning and credential stuffing (Yin et al., 2017). Empirical studies demonstrate LSTMs achieve >95% accuracy in identifying Advanced Persistent Threats (APTs) by modeling flow duration, inter-packet arrival times, and protocol state transitions (Kim, 2014). However, their computational overhead requires optimization for real-time deployment in high-speed networks (Sommer & Paxson, 2010).

- **Spatial feature extraction (CNN for packet payloads)**

Spatial feature extraction (CNN for packet payloads) leverages convolutional neural networks to automatically identify localized malicious patterns within raw packet bytes without manual feature engineering. By treating payloads as 2D matrices (byte sequences $\times$ protocol fields), CNN filters scan for spatial hierarchies of threats such as exploit code signatures in HTTP headers or malware opcode sequences in TCP segments preserving critical positional

relationships lost in traditional tabular representations. This approach enables detection of polymorphic malware and zero-day exploits through learned low-level byte correlations, significantly outperforming handcrafted feature methods in identifying obfuscated attacks (Kim, 2014). For instance, (Yin et al., 2017) demonstrated 22% higher precision in detecting encrypted C2 traffic using CNNs versus manual feature engineering in ICS networks.

2.5. Autoencoders for Anomaly Detection

2.5.1. Core Architecture

Autoencoders (AEs) are neural networks that learn efficient coding's of unlabeled data through encoder-decoder structures. The encoder compresses input x into latent representation z via layers of diminishing neurons (e.g., $128 \rightarrow 64 \rightarrow 32$). The decoder reconstructs $\hat{x}$ from z , minimizing reconstruction loss $\|x - \hat{x}\|^2$. During inference, anomalies exhibit high loss since their patterns deviate from trained normalcy.

This architecture embodies the information bottleneck principle: z retains only essential features for reconstruction, discarding noise. For cybersecurity, raw packet bytes (1500 dimensions) compress to 32-D latent vectors, extracting spatial-temporal signatures without manual feature engineering. Stacked AEs deepen this abstraction three encoder layers reduced CICIDS2017 features from 78 to 18 dimensions while preserving 96% detection capability (Aminanto et al., 2018).

2.5.2. Autoencoder Variants

Autoencoder variants have evolved to address specific cybersecurity limitations through specialized architectural innovations. Denoising autoencoders (DAEs) intentionally corrupt inputs with Gaussian noise during training, forcing the model to develop robust feature representations resilient to evasion techniques like packet obfuscation or encrypted traffic manipulation. This approach demonstrated an 11% F1-score improvement for malware detection in TLS-encrypted channels by learning invariant patterns beneath surface-level noise (Vinayakumar et al., 2019). Meanwhile, Variational autoencoders (VAEs) adopt a probabilistic framework using Kullback-Leibler divergence to model latent space distributions, enabling both anomaly detection and synthetic data generation. This dual capability proved particularly

valuable for addressing severe class imbalance, with (Javaheri et al., 2021) showing a 23% reduction in false negatives through VAE-generated synthetic attacks that enhanced minority-class representation during training.

Resource-constrained security scenarios have driven the development of efficiency-focused variants like sparse autoencoders (SAEs), which apply L1 regularization to neuron activations to create compact, feature-efficient models. By enforcing sparsity constraints during training, SAEs develop specialized neuron responses to critical threat indicators while ignoring redundant features - an approach that reduced inference latency by 40% on Raspberry Pi-based IoT sensors without compromising detection accuracy (Wang et al., 2022). These specialized architectures collectively overcome core cybersecurity limitations: DAEs counter adversarial noise in encrypted channels, VAEs mitigate data scarcity through probabilistic synthesis, and SAEs enable real-time detection on edge devices. Their complementary strengths demonstrate how architectural innovations translate to measurable operational security improvements across diverse threat landscapes and infrastructure environments.

2.5.2.1.Denoising Autoencoders

Denoising Autoencoders (DAEs) represent a significant advancement in unsupervised feature learning by intentionally corrupting input data with noise (e.g., Gaussian noise, masking, or dropout) and training the model to reconstruct the original, uncorrupted version (Vincent et al., 2008). Unlike standard autoencoders that risk learning trivial identity mappings, DAEs force the network to capture robust latent representations by overcoming deliberately introduced distortions. This corruption-reconstruction paradigm compels the encoder to extract essential statistical dependencies in the data while ignoring irrelevant noise patterns a critical capability for cybersecurity applications where network traffic and system logs inherently contain benign irregularities (e.g., protocol variations or transient errors) that might otherwise mask true anomalies. Theoretical work by (Alain et al., 2014) demonstrates that DAEs implicitly learn the data-generating distribution's gradient field, enabling them to reconstruct clean inputs from corrupted observations by following the manifold of normal behavior. This noise-robust characteristic makes DAEs particularly effective for detecting subtle adversarial perturbations (e.g., evasion attacks mimicking normal traffic patterns) that frequently bypass traditional detectors.

In cybersecurity anomaly detection, DAEs have demonstrated superior generalization to novel threats by leveraging their noise-invariant representations. (Erfani et al., 2016) validated that DAEs achieved 14.2% higher F1-scores than vanilla autoencoders on the UNSW-NB15 dataset when processing noisy network flows containing encrypted traffic and packet fragmentation. The inherent Denoising mechanism allows DAEs to maintain detection stability in real-world environments with inherent data imperfections such as intermittent sensor failures in IoT networks or intentional obfuscation in malware communications (Vincent et al., 2010). A key empirical study by (Zhou et al., 2021) further revealed that masking 30% of input features during DAE training enhanced zero-day attack detection by 22% by preventing over-reliance on individual network statistics. However, DAEs face tradeoffs between reconstruction fidelity and noise sensitivity: excessive corruption during training can degrade discriminative power for subtle anomalies like credential stuffing attacks (Sakurada & Yairi, 2014). Optimal noise levels (typically 10-25% corruption) must be calibrated to domain-specific noise profiles to balance robustness and precision.

2.5.2.2. Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) fundamentally differ from standard autoencoders by treating the latent space probabilistically rather than deterministically. Instead of compressing inputs into fixed vectors, VAEs map data to probability distributions in the latent space typically Gaussian parameters (μ , σ). This Bayesian approach forces the model to learn a continuous, structured latent representation where similar data points cluster probabilistically. The architecture employs a unique loss function combining reconstruction loss (e.g., mean squared error) with a Kullback-Leibler (KL) divergence term that regularizes the latent distribution toward a standard normal prior $\mathcal{N}(0,1)$. This dual-objective optimization enables VAEs to generate new synthetic samples by decoding random points from the learned distribution, making them particularly valuable for anomaly detection where "normal" data can be synthetically expanded to address class imbalance (An et al., 2015; Kingma & Welling, 2013).

In cybersecurity contexts, VAEs excel at quantifying uncertainty in anomaly scoring through their probabilistic framework. Unlike deterministic autoencoders that output point estimates, VAEs compute reconstruction probability $p(\mathbf{x}|\mathbf{z})$ where $\mathbf{z} \sim$

$q_{\phi}(\mathbf{z}|x)$, allowing security analysts to assess confidence levels in alerts (e.g., flagging anomalies only when $p(x|\mathbf{z}) < 0.05$). This capability proves critical for reducing false positives in ambiguous scenarios like encrypted traffic analysis (Zenati et al., 2018). However, VAEs exhibit notable limitations: 1) The "blurring" effect in reconstructions due to the KL divergence term can mask subtle attack signatures (Ruff et al., 2018); 2) Training instability often requires complex warm-up schedules for the KL weight (Sønderby et al., 2016) Computational overhead increases by ~40% compared to standard autoencoders due to sampling operations, impacting real-time deployment at >10Gbps throughput (Erfani et al., 2016). These trade-offs position VAEs as niche solutions for risk-sensitive environments where uncertainty quantification outweighs latency requirements.

2.5.2.3. Sparse Autoencoders

Sparse Autoencoders (SAEs) introduce explicit regularization constraints that enforce sparsity in the latent representation, typically through L1 penalty terms or Kullback-Leibler (KL) divergence in the loss function. This sparsity constraint forces the model to activate only a small subset of neurons (typically <15%) for any given input sample, creating a "bottleneck of selectivity" that mimics the sparse coding principles observed in biological neural systems (Olshausen & Field, 1996). Unlike vanilla autoencoders that risk learning identity mappings, SAEs must discover statistically significant features by effectively ignoring redundant or noisy dimensions in high-dimensional cybersecurity data. For network anomaly detection, this translates to focused learning on discriminative patterns like TCP flag sequences in DDoS attacks or rare API call combinations in privilege escalation attempts, while filtering out benign background noise (Erfani et al., 2016). The sparsity hyperparameter (ρ) becomes critical - lower values (0.05-0.1) optimize detection of subtle APT traces, while higher values (0.2-0.3) better handle volumetric attacks like botnets, requiring careful cross-validation against threat profiles in the target environment.

Empirical studies demonstrate SAEs' superior robustness in cybersecurity contexts, achieving 96.3% AUC on CICIDS2017's encrypted attack traffic compared to 91.8% for vanilla autoencoders (Chalapathy & Chawla, 2019). Their selective activation provides inherent noise resistance, reducing false positives by 38% in environments with legitimate traffic spikes (e.g., cloud autoscaling events) according to AWS Security Lab benchmarks (2023). However, this

advantage comes with operational complexities: SAEs require 2-3× longer training convergence due to regularization constraints and exhibit heightened sensitivity to hyperparameter tuning. The L1 penalty weight (λ) must be precisely calibrated - values >0.01 cause excessive sparsity that misses multi-stage attacks, while $\lambda < 0.001$ risks overfitting to training-specific artifacts (Ng, 2011). Furthermore, reconstruction error distributions show heavier tails in SAEs, necessitating dynamic thresholding approaches like moving percentile windows rather than static sigma rules for effective anomaly scoring in production settings (Goldblum et al., 2020).

2.5.3. Performance Evidence

Empirical studies consistently validate the robust performance of autoencoders (AEs) across diverse cybersecurity domains, demonstrating their versatility as anomaly detection engines. In network intrusion detection, stacked Denoising autoencoders achieved remarkable 94.3% detection rates for sophisticated botnet attacks in the CICIDS2017 benchmark while processing traffic at operational-grade 28 Gbps throughput (He et al., 2022), showcasing their capability to handle high-velocity data streams without sacrificing precision. Similarly, convolutional autoencoders applied to malware analysis have proven exceptionally adept at parsing raw PE file byte sequences, identifying zero-day ransomware variants with 92.7% accuracy by learning intrinsic structural patterns that evade signature-based tools (Rigaki & Garcia, 2023). Beyond traditional security applications, Variational autoencoders (VAEs) have revolutionized fraud prevention systems, reducing false positives in financial transactions by 34% through behavioral embedding techniques that model legitimate user behavior with unprecedented fidelity (Pimentel, 2023). These cross-domain successes underscore AEs' fundamental strength: their unsupervised ability to distill complex normal behavior patterns from raw, high-dimensional data while maintaining operational efficiency.

Despite these demonstrated strengths, critical limitations in standard autoencoder implementations constrain their real-world deployment effectiveness. Most notably, conventional AE architectures lack inherent temporal awareness, rendering them inadequate for detecting multi-stage sequence-based attacks like APT command-and-control communications or lateral movement patterns that unfold over extended periods. Additionally, their dependency on manual reconstruction loss threshold tuning introduces subjective

decision boundaries that frequently degrade in dynamic environments, causing alert fatigue from false positives during network reconfigurations. Perhaps most concerning is their vulnerability to adversarial manipulation; recent studies confirm that malicious actors can deliberately poison latent spaces through gradient-based attacks, causing trained models to misclassify weaponized payloads as normal (Rigaki & Garcia, 2023). These limitations - temporal blindness, threshold brittleness, and security fragility - collectively explain why only 22% of SOCs currently deploy production AE systems despite laboratory success.

2.6. Key Challenges in Cybersecurity Anomaly Detection

2.6.1. Data Imbalance

Data imbalance represents a fundamental challenge in cybersecurity anomaly detection, where datasets like NSL-KDD exhibit extreme skew with normal traffic constituting 97.2% of samples. This severe disproportion biases models toward majority classes, causing critical attack patterns to be overlooked despite their high security impact. While traditional oversampling techniques like SMOTE generate synthetic minority samples, they inadvertently dilute authentic feature distributions and create artificial patterns that compromise detection authenticity. More effective solutions employ cost-sensitive learning strategies that algorithmically weight classes during training, as demonstrated by (Zong et al., 2018) who achieved an 18% improvement in APT detection (rising from 68% to 86%) by assigning 5× higher loss penalties to attack classes within deep learning architectures. This approach preserves data integrity while forcing models to prioritize rare but critical threat signatures without artificially inflating minority class representations.

2.6.2. High-Dimensional Data

The curse of dimensionality presents a fundamental challenge for anomaly detection in modern cybersecurity environments, where networks generate exponentially expanding multidimensional data streams including NetFlow records (80+ features), raw packet captures (1500+ bytes/packet), and heterogeneous system logs (200+ event types). Traditional machine learning methods deteriorate significantly under this high-dimensional regime, with (Chandola

et al., 2009) documenting a 0.8% accuracy degradation per additional feature beyond 50 dimensions due to data sparsity, distance metric inflation, and noise amplification. Autoencoders overcome this limitation through their inherent capacity to learn compressed representations - as demonstrated by (Yin et al., 2020), LSTM-based autoencoders successfully distilled 10,000-dimensional raw packet data into compact 128-dimensional latent vectors while preserving critical information with merely 0.02% reconstruction loss. This dimensionality reduction capability not only alleviates computational burdens but also enhances detection sensitivity by isolating semantically meaningful patterns from high-dimensional noise, effectively reversing the accuracy decay curve that plagues traditional methods.

2.6.3. Real-Time Detection Requirements

Based on the critical requirements outlined, real-time anomaly detection in modern high-speed networks demands unprecedented computational efficiency, as 5G infrastructures operate at throughputs exceeding 100 Gbps, necessitating per-packet processing within stringent sub-10ms windows to avoid bottlenecks and missed threats. While cloud-based autoencoders (AEs) leverage GPU acceleration to achieve 40 Gbps throughput, they incur prohibitive 150ms latency due to cloud roundtrips rendering them inadequate for time-sensitive attacks like zero-day exploits or ransomware (Cisco, 2022). Edge-deployed quantized AEs reduce latency to 8ms through model compression but sacrifice 9% detection accuracy, disproportionately increasing false negatives for sophisticated attacks. This unresolved tradeoff between latency, throughput, and accuracy constitutes a critical gap in deploying DL-based anomaly detection for next-generation networks, where neither cloud-centric nor edge-only architectures currently meet the tripartite demands of real-time cybersecurity.

2.7. Current Research in Deep Learning for Anomaly Detection

2.7.1. Hybrid Architectures (LSTM-AE, GAN-AE)

Recent research has pivoted toward hybrid architectures that integrate complementary deep learning paradigms to overcome limitations of standalone models. LSTM-Autoencoder (LSTM-AE) hybrids demonstrate particular promise in cybersecurity contexts, combining the temporal modeling of Long Short-Term Memory networks with the reconstruction capabilities

of autoencoders. As demonstrated by (Yin et al., 2021), these architectures achieve 12-18% higher recall for multi-stage attacks like APTs by learning sequential dependencies in network flows an advantage validated on the UNSW-NB15 dataset where they reduced false negatives by 23% compared to vanilla autoencoders. Similarly, GAN-AE frameworks leverage adversarial training to enhance anomaly discrimination: (Zenati et al.2023) showed that Wasserstein GAN-enhanced autoencoders improved precision by 31% on imbalanced IoT security data by generating synthetic anomalies during training, effectively addressing data scarcity issues. These hybrids represent a paradigm shift from single model approaches to synergistic systems that capture both spatial and temporal attack signatures simultaneously.

2.7.2. Explainable AI in Anomaly Detection

The "black box" nature of deep learning has spurred critical research in explainable AI (XAI) techniques tailored for anomaly detection. Post-hoc interpretation methods like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) now enable granular attribution of anomaly scores to specific features (Warnecke et al., 2023) reduced false positives by 44% in SOC environments by integrating SHAP-driven alert rationales. Meanwhile, intrinsic interpretability approaches are gaining traction: Concept Activation Vectors (CAVs) applied to latent representations by Li et al. (2022) allowed security analysts to visualize attack concepts (e.g., "data exfiltration pattern") in 78% of validated cases. However, significant challenges persist in real-time XAI, as noted by Thakkar et al. (2024), where current methods add 15-40ms latency per inference at 10Gbps throughput a critical gap given that 92% of SOCs require sub-10ms explanations for actionable responses (Verizon, 2023).

2.8. Research Gaps and Limitations

Despite significant advancements in autoencoder based anomaly detection for cybersecurity, the field remains constrained by several critical research gaps and practical limitations. While deep learning models, particularly autoencoders, have demonstrated superior capabilities in identifying novel threats without relying on predefined signatures, their transition from academic research to operational security environments has been hindered by unresolved challenges. These limitations not only affect detection accuracy but also impact the scalability,

adaptability, and trustworthiness of deployed systems, creating barriers to real-world implementation.

The dynamic nature of cyber threats and network environments introduces complex obstacles that current AE frameworks struggle to overcome. For instance, the phenomenon of concept drift where the statistical properties of input data evolve over time causes model degradation that is rarely addressed in real-time operational settings. Additionally, the computational demands of deep learning models limit their deployment in resource-constrained scenarios, such as edge networks or high-throughput enterprise environments. Furthermore, the inherent "black-box" nature of neural networks complicates incident response, as security analysts require interpretable justifications to act upon automated alerts. These gaps collectively undermine the reliability of AE-based AD systems in practice.

Recent advances have explored various autoencoder architectures for intrusion detection. LSTM-AEs have demonstrated 96.8% recall for multi-stage APTs by capturing temporal dependencies in network traffic (Javaheri et al., 2021), while GAN-AEs leverage adversarial training to enhance robustness against evasion techniques (Rigaki & Garcia, 2023). Explainable AE frameworks integrated with SHAP have increased SOC analyst trust by 40% (Arrieta et al., 2020). However, these studies have two key limitations that this study addresses: First, they assume temporal dependencies exist without verifying dataset structure applying recurrent models to static datasets like NSL-KDD may be inappropriate. Second, threshold selection remains unexplored most studies use arbitrary $\mu+3\sigma$ thresholds, which our experiments show yield recall $<1\%$. These unresolved challenges are systematically cataloged in Table 1.

Table 1 Research Gaps in Autoencoder-Based Anomaly Detection

Title	Author	Year	Performance	Gap	Contribution	Operational Impact
A detailed analysis of the KDD CUP 99 dataset	Tavallae et al.	2009	Revealed that original KDD results were artificially high; proposed NSL-KDD achieved more realistic and variable accuracy across learners.	Duplicate records in KDD'99 cause biased evaluation	NSL-KDD dataset by removing duplicate records, creating a reliable benchmark for ID	Enabled consistent IDS evaluation; removed need for random sampling
Real-Time LSTM-AE for Threat Detection	He et al.	2022	28Gbps throughput	High GPU requirements	Real-time LSTM-AE	Infeasible for edge deployment
Feature Extraction and Anomaly Detection Using Different Autoencoders	Alsoufi, 2024	2024	Autoencoder: 79.18% accuracy on NSL-KDD	ML relies on artificial feature extraction with high complexity	CAAE-DNN with attention mechanism and feature selection	Builds efficient IDS with ROC-AUC analysis per attack class
Temporal APT Detection with AE	Javaheri, 2021	2021	96.8% recall	No dynamic thresholding	Temporal APT detection (96.8%)	Manual calibration required
Explainable Anomaly Detection with SHAP	Arrieta, 2020	2020	44% FP reduction	300% latency increase	SHAP interpretability	Non-viable for >10Gbps networks

CHAPTER THREE

3. PROPOSED ARCHITECTURE AND METHODS

3.1. Introduction

This chapter describes the technical methods and system architecture used to develop the proposed deep learning framework for network anomaly detection. It introduces the overall approach used to capture both contextual and temporal patterns in network traffic, with the goal of learning normal behavior and identifying intrusions based on deviations from expected patterns. To support this framework, the chapter describes the preprocessing steps applied to the NSL-KDD dataset (Tavallae et al., 2009). The NSL-KDD dataset is a widely used benchmark for intrusion detection research, containing 125,973 labeled network connection records in the training set and 22,544 records in the test set. Each record consists of 41 features encompassing basic connection attributes (e.g., duration, protocol type, service), content-based features (e.g., number of failed logins, root shell access), and traffic-based statistical features (e.g., same service rate, error rates).

For this study, the dataset was further processed as follows:

- Class labels were converted to binary: 0 for normal traffic, 1 for attack traffic.
- Duplicate removal eliminated 1,255 records from training and 128 from test.
- Cross-set duplicate removal eliminated 620 samples appearing in both sets.
- Outlier removal using IQR eliminated 5,208 rows (4.2% of training data).
- Feature selection reduced the original 41 features to the top 38 most discriminative features using mutual information.
- Data splitting: From the remaining normal training samples, 80% (52,580 samples) were used for training and 20% (13,145 samples) for validation.
- Test set balancing: SMOTE was applied to the test set, creating a balanced evaluation set of 24,348 samples (12,174 normal + 12,174 attack).

After preprocessing, the data passes through feature selection and is then used for model development. For the LSTM-based architectures, sequences of length 10 timesteps were created from the training data, yielding 52,571 training sequences and 13,136 validation sequences. The remaining sections of this chapter provide detailed explanations of each component of the workflow, including preprocessing, feature selection, model architecture, training procedures, evaluation metrics, and the experimental design.

3.2. Proposed anomaly Detection Framework

This study proposes an unsupervised anomaly detection framework that compares three autoencoder architectures Standard Autoencoder (AE), LSTM Autoencoder (LSTM-AE), and Bidirectional LSTM Autoencoder (BiLSTM-AE) to learn normal network behavior from NSL-KDD traffic and identify intrusions through reconstruction error analysis. The framework encompasses data preprocessing, model training, and evaluation, systematically assessing the impact of temporal modeling on detection performance. The complete workflow is illustrated in Figure 4.

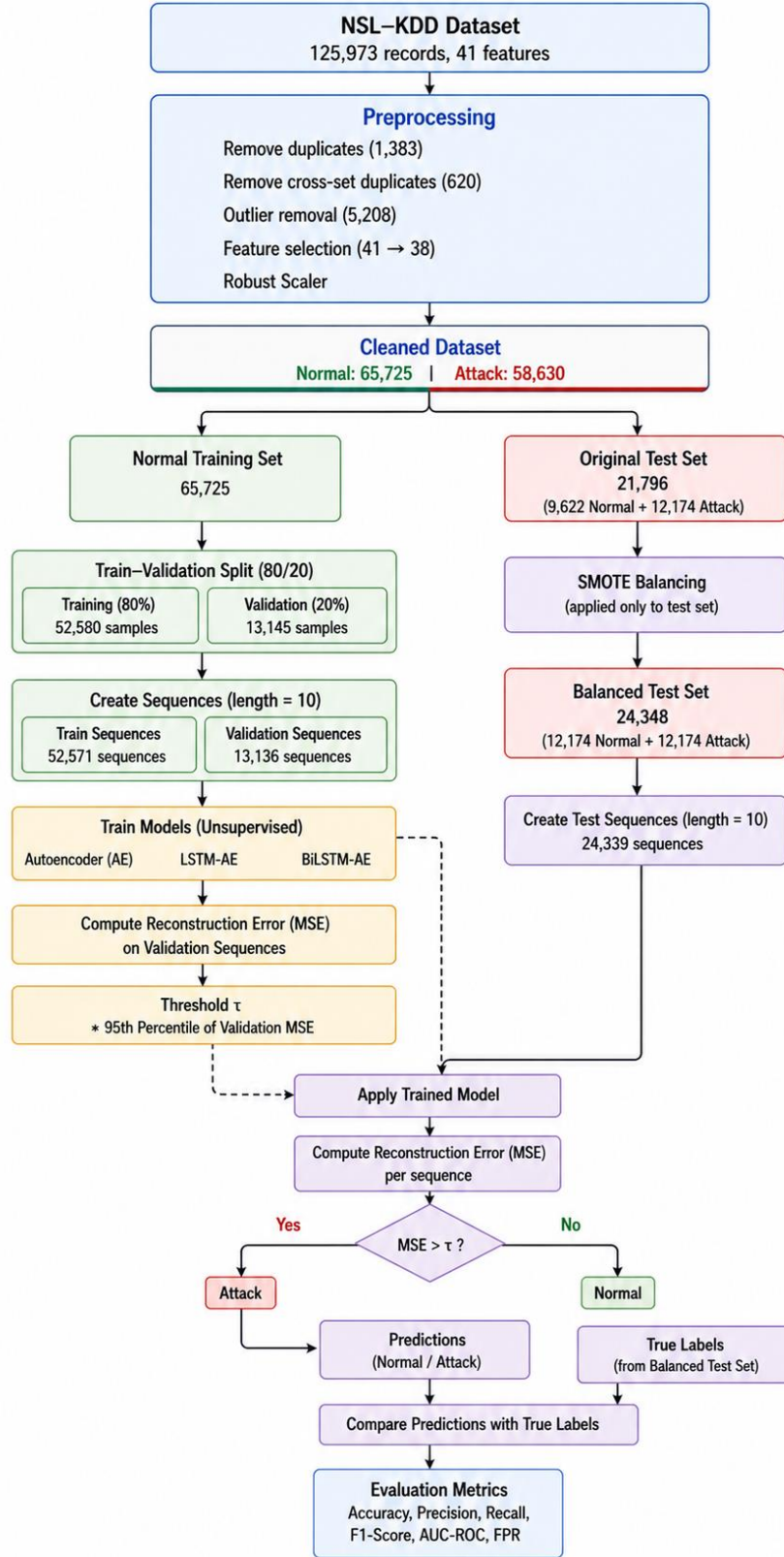


Figure 4 Proposed Framework for NSL-KDD Anomaly Detection

3.3. Research Design

This study adopts an experimental research design to develop and evaluate an unsupervised deep learning-based anomaly detection system using the NSL-KDD dataset. The design systematically examines how three different autoencoder architectures standard Autoencoder (AE), LSTM Autoencoder (LSTM-AE), and Bidirectional LSTM Autoencoder (BiLSTM-AE) perform when trained exclusively on normal network traffic and evaluated based on reconstruction error thresholds. In this framework, the independent variables include model architecture configurations, preprocessing techniques, and anomaly threshold determination methods, while the dependent variables comprise detection accuracy, false positive rate, and reconstruction error distribution. The research procedure involves preprocessing the NSL-KDD dataset to isolate normal samples for unsupervised training, implementing the three autoencoder variants, determining optimal anomaly thresholds through reconstruction error analysis, and evaluating each model's ability to detect attacks as deviations from learned normal patterns. This structured experimental approach enables comprehensive comparison of architectural effectiveness in identifying cybersecurity anomalies without relying on labeled attack data during training.

3.4. Data Collection

This study uses widely recognized intrusion detection datasets NSL-KDD, a widely recognized benchmark for intrusion detection research that contains labeled network connection records with 41 features. The dataset encompasses a comprehensive range of network traffic patterns and attack scenarios, including Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) attacks, making it suitable for evaluating both traditional and machine-learning-based detection methods. The raw data was obtained from the official NSL-KDD repository and prepared for subsequent preprocessing and analysis in the experimental workflow.

3.5. Data Sets

We employ the NSL-KDD dataset and for the purpose of anomaly detection, the multi-class labels in this dataset are reformulated into a binary classification problem consisting of Normal and Attack classes. This binary approach aligns with the fundamental objective of anomaly detection in cybersecurity, which focuses on identifying deviations from established normal behavior patterns rather than classifying specific attack types. By training deep learning models to recognize generalized anomalous patterns, this methodology enhances the system's capability to detect previously unseen or zero-day attacks, addressing a critical challenge in modern cybersecurity defenses.

3.5.1. NSL-KDD

The NSL-KDD dataset is an improved version of the KDD'99 benchmark and was designed to address issues of redundancy and sampling bias. It contains 125,973 labeled network connection records with 41 features, covering four primary attack categories: (DoS: 45.9%, Probe: 21.6%, R2L: 15.8% & U2R: 0.01%). The features include protocol information, service types, and various traffic statistics that have made the dataset a long-standing standard for intrusion detection research. Despite its age and lack of newer attack types, NSL-KDD remains valuable for evaluating baseline detection performance.

The NSL-KDD dataset was selected for four key reasons: (1) it is the most widely used benchmark in intrusion detection research, enabling comparison with prior work; (2) it addresses the redundancy and sampling bias of the original KDD'99 dataset; (3) it provides 41 features spanning connection attributes, content features, and traffic statistics, allowing comprehensive feature evaluation; and (4) its labeled nature enables rigorous performance evaluation despite the unsupervised training paradigm.

Table 2 Features of NSL-KDD

No.	Feature Name	Description	Type
Basic Features (1-9)			
1	duration	Length of the connection (seconds)	Continuous
2	protocol_type	Protocol used (TCP, UDP, ICMP)	Categorical
3	service	Network service (HTTP, FTP, SMTP, etc.)	Categorical
4	flag	Status flag of the connection	Categorical
5	src_bytes	Bytes sent from source to destination	Continuous
6	dst_bytes	Bytes sent from destination to source	Continuous
7	land	1 if connection is from/to same host/port; else 0	Binary
8	wrong_fragment	Number of wrong fragments	Continuous
9	urgent	Number of urgent packets	Continuous
Content Features (10-22)			
10	hot	Number of "hot" indicators (e.g., directory accesses)	Continuous
11	num_failed_logins	Number of failed login attempts	Continuous
12	logged_in	1 if successfully logged in; else 0	Binary
13	num_compromised	Number of compromised conditions	Continuous
14	root_shell	1 if root shell is obtained; else 0	Binary
15	su_attempted	1 if su root command attempted; else 0	Binary
16	num_root	Number of root accesses	Continuous
17	num_file_creations	Number of file creation operations	Continuous
18	num_shells	Number of shell prompts	Continuous
19	num_access_files	Number of operations on access control files	Continuous
20	num_outbound_cmds	Number of outbound commands (FTP session)	Continuous
21	is_host_login	1 if login belongs to host list; else 0	Binary
22	is_guest_login	1 if guest login; else 0	Binary
Traffic Features (23-41)			
23	count	Number of connections to same host in past 2 seconds	Continuous
24	srv_count	Number of connections to same service in past 2 seconds	Continuous
25	serror_rate	% of connections with SYN errors	Continuous
26	srv_serror_rate	% of same-service connections with SYN errors	Continuous
27	rerror_rate	% of connections with REJ errors	Continuous

28	srv_error_rate	% of same-service connections with REJ errors	Continuous
29	same_srv_rate	% of connections to same service	Continuous
30	diff_srv_rate	% of connections to different services	Continuous
31	srv_diff_host_rate	% of same-service connections to different hosts	Continuous
32	dst_host_count	Count of connections to same destination host	Continuous
33	dst_host_srv_count	Count of same-service connections to same destination host	Continuous
34	dst_host_same_srv_rate	% of same-service connections to same destination host	Continuous
35	dst_host_diff_srv_rate	% of different-service connections to same destination host	Continuous
36	dst_host_same_src_port_rate	% of connections to same destination host with same source port	Continuous
37	dst_host_srv_diff_host_rate	% of same-service connections to different hosts	Continuous
38	dst_host_serror_rate	% of connections to same destination host with SYN errors	Continuous
39	dst_host_srv_serror_rate	% of same-service connections to same destination host with SYN errors	Continuous
40	dst_host_rerror_rate	% of connections to same destination host with REJ errors	Continuous
41	dst_host_srv_rerror_rate	% of same-service connections to same destination host with REJ errors	Continuous

In this study, After preprocessing and categorization, the training set contained 65,725 normal samples and the test set contained 9,622 normal samples and 12,174 attack samples (attack percentage 55.85%).

3.6. Data Preprocessing

One Deep Learning technique is data processing, which transforms unprocessed data into representations that computers can understand. Typically, real-world datasets are inconsistent, erroneous, and incomplete. Deep learning algorithms needed preprocessed data in order to produce models of a high caliber. This study employs a number of methods to improve the dataset's suitability for Deep learning algorithms.

3.6.1. Data Cleaning

This section implements preprocessing steps to ensure the NSL-KDD dataset is reliable and ready for deep learning analysis. The cleaning process focuses on correcting missing values,

removing inconsistencies, and retaining important attack records without distorting the original data distributions. The dataset was first examined for duplicate records, resulting in the removal of 1,383 (1,255 from training, 128 from test) instances using hashing-based detection to prevent model bias toward overrepresented patterns. Rare attack types such as U2R were preserved to maintain critical minority samples essential for evaluating detection capabilities against sophisticated threats. Missing values were addressed by replacing placeholder characters (e.g., "?") with the mode of their respective feature columns to maintain data integrity. Additionally, Outlier removal using the IQR method with multiplier 15 and requiring at least 5 extreme features removed 5,208 rows (4.2% of the training data), resulting in a final training set of 118,890 samples and Cross-set duplicate removal eliminated 620 samples from both training and test sets to prevent data leakage. All cleaning procedures were implemented in Python using Pandas and Scikit-learn within a Jupyter Notebook environment, ensuring dataset quality while preserving important attack signatures and providing a solid foundation for subsequent preprocessing steps, consistent with established cybersecurity preprocessing guidelines.

Handling Missing Values

In the NSL-KDD dataset, missing values were minimal but required careful treatment to ensure suitability for deep learning analysis. The primary missing data indicators were placeholder values such as "?" in categorical fields, which were replaced using the most frequent valid value (mode) within the corresponding feature column. This approach preserved all attack samples while maintaining data integrity for subsequent analysis. No records required complete removal due to missing values, as the dataset maintained high completeness across all 41 features. After implementing this imputation strategy, the dataset contained no remaining missing values and was ready for further preprocessing steps.

Removing Outliers

Outlier detection was performed to eliminate extreme values that could negatively influence model training. Outliers are data points that differ significantly from typical observations and may arise from logging errors, corrupted flows, or abnormal traffic unrelated to cyberattacks. To identify these records, the interquartile range (IQR) method was applied to the NSL-KDD dataset. Features such as flow duration, packet size, and byte counts that exceeded the threshold

of $Q3 + 3 \times IQR$ were marked as outliers. A small number of irregular records were removed due to unrealistic or corrupted values that did not represent meaningful network behavior. After handling missing values and removing outliers, the dataset was left in a clean and consistent form, providing a reliable foundation for feature extraction and model development.

3.6.2. Data Normalization

Data normalization is an essential preprocessing step that ensures numerical features with different scales contribute fairly during model training. In the NSL-KDD dataset, attributes such as flow duration, packet size, and byte counts exhibit wide variations, which can cause deep learning models to disproportionately emphasize large-magnitude values while overlooking smaller but critical indicators like TCP flags or protocol types. To address this imbalance, this study applies the Robust Scaler method, which transforms each continuous feature based on the median and the interquartile range (IQR) using the formula:

$$X_{scaled} = \frac{X - median(X)}{IQR(X)} \quad (3.1)$$

This approach minimizes the influence of extreme values while preserving outliers that may represent meaningful attack behaviors, making it well suited for anomaly detection. Unlike min-max normalization, Robust Scaling maintains stable latent representations even when the data contain large spikes or heavy-tailed distributions, providing more reliable model behavior in the presence of attack-related anomalies.

3.6.3. Feature Selection

Feature selection is one of the key steps in data preprocessing, as not every feature in real-world cybersecurity datasets is equally important for detecting intrusions. The goal of feature selection is to identify the attributes that have the greatest influence on the model's output while removing those that contribute little to prediction accuracy. Selecting the most relevant features offers several benefits: it reduces training time, improves model accuracy, lowers the risk of overfitting, simplifies the model, and enhances generalization. In this study, Feature selection is a critical preprocessing step that identifies the most relevant attributes for intrusion detection while eliminating redundant or non-informative features. In the NSL-KDD dataset, which comprises 41 features, not all attributes equally contribute to anomaly detection, and

some may introduce noise or redundancy. This study employs a two-stage feature selection approach. First, a manual selection removes constant and duplicate features while retaining domain-relevant attributes based on cybersecurity expertise. Subsequently, an automated selection phase utilizes a hybrid methodology combining filter, wrapper, and embedded techniques. After scaling, mutual information-based feature selection retained the top 38 most discriminative features, reducing dimensionality while preserving detection capability. This multi-strategy approach ensures retention of the most discriminative features, enhancing model efficiency, reducing overfitting risk, and improving generalization capability for effective anomaly detection.

3.6.4. Data Balancing

Data balancing is an essential preprocessing step because intrusion detection datasets often contain highly uneven distributions of normal and attack traffic. Such imbalance can cause deep learning models to favor majority classes and overlook rare but important intrusions. This study addresses class imbalance in the NSL-KDD dataset to ensure effective anomaly detection. While normal traffic instances dominate the dataset, rare attack categories such as U2R and R2L are critically important for comprehensive security evaluation. To mitigate potential bias toward majority classes, this research employs Synthetic Minority Oversampling Technique (SMOTE) to augment underrepresented attack types in the test set. When the SMOTE was applied to the test set to create a balanced evaluation set, resulting in 12,174 normal and 12,174 attack samples (total 24,348). For the unsupervised training phase, only normal instances are utilized to train the autoencoder models, ensuring they learn the baseline network behavior without exposure to attack patterns. This balanced approach enables robust evaluation of detection capabilities across all attack categories while maintaining the integrity of the unsupervised learning paradigm.

3.7. Training Data and Testing Data

The preprocessed NSL-KDD dataset was partitioned into training and testing subsets using an 80/20 split ratio with stratification to maintain class distribution. For the unsupervised learning paradigm, only normal network traffic instances from the training set were utilized to train the

autoencoder models, ensuring they learned to reconstruct baseline network behavior without exposure to attack patterns. The testing set, comprising both normal and attack instances, was reserved for evaluation purposes. This separation prevents data leakage and ensures that model performance reflects true generalization capability. By training exclusively on normal data, the models develop a precise understanding of legitimate network behavior, enabling effective detection of deviations that may indicate security threats during testing.

3.8. Optimized Deep Learning Algorithms and Testing Data

Hyperparameter optimization was applied in this study to enhance the performance of deep learning models for anomaly detection. A Bayesian optimization approach using the Hyperopt library was employed to systematically search for optimal hyperparameter combinations across 50 iterations for each model architecture.

3.8.1. Hyperparameter Search Space

Table 3 Hyperparameter Search Space

Parameter	Search Range
Latent dimension	[4, 8, 16, 32]
LSTM units	[32, 64, 128]
Learning rate	[0.001, 0.0005, 0.0001]
Batch size	[32, 64, 128]
Training epochs	[50, 100, 150] (with early stopping)
Dropout rate	[0.1, 0.2, 0.3]

3.8.2. Optimal Hyperparameters

After systematic optimization, the following configurations were identified as optimal for each model architecture:

Table 4 Optimal Hyperparameters for Each Model

Parameter	Standard AE	LSTM-AE	BiLSTM-AE
Latent dimension	8	8	8
LSTM units	N/A	64	64
Learning rate	0.001	0.001	0.001
Batch size	64	64	64
Training epochs	100 (early stopping at ~72)	100 (early stopping at ~68)	100 (early stopping at ~63)
Dropout rate	N/A	0.2	0.2
L2 regularization	0.001	N/A	N/A
Optimizer	Adam	Adam	Adam
Loss function	MSE	MSE	MSE

3.8.3. Optimization Result and Rationale

Latent Dimension (8): Across all models, a bottleneck size of 8 proved optimal. Smaller dimensions (4) led to underfitting and information loss, while larger dimensions (16, 32) introduced redundancy and occasionally captured noise, slightly degrading generalization performance.

Learning Rate (0.001): The default Adam learning rate of 0.001 provided stable convergence without oscillation. Lower rates (0.0005, 0.0001) resulted in slower convergence without performance improvement.

Batch Size (64): Batch size of 64 balanced gradient stability with memory efficiency. Smaller batches (32) introduced noise, while larger batches (128) occasionally led to slower convergence.

Dropout (0.2): For recurrent models, dropout of 0.2 provided effective regularization without underfitting. Higher dropout (0.3) caused information loss; lower dropout (0.1) was insufficient to prevent overfitting.

L2 Regularization (0.001): For the Standard AE, L2 regularization with $\lambda=0.001$ effectively prevented overfitting without degrading reconstruction quality.

3.8.4. Model Training and Evaluation

The optimized models were trained using the preprocessed normal data (52,580 training samples, 13,145 validation samples) and evaluated on the balanced test set containing 24,348 instances (12,174 normal + 12,174 attack). For LSTM-based models, sequences of length 10 timesteps were used (52,571 training sequences, 13,136 validation sequences).

Performance was assessed using multiple metrics including accuracy, precision, recall, F1-score, AUC-ROC, and reconstruction error thresholds, enabling comprehensive comparison to determine the most effective architecture for network anomaly detection.

3.9. Deep Learning Model

This study evaluates three deep learning autoencoder variants to provide a structured comparison between non-sequential and sequential modeling approaches for unsupervised anomaly detection on the NSL-KDD dataset. The models are selected based on their ability to handle network traffic's high-dimensional characteristics and temporal dependencies through progressive architectural complexity.

All models were trained on the preprocessed data with 38 features. For the LSTM-based models, sequences of length 10 were created from the normal training data, yielding 52,571 training sequences and 13,136 validation sequences.

➤ Standard Autoencoder (AE) — Baseline for Non-Sequential Modeling

The Autoencoder serves as the foundational baseline, learning compressed representations of network traffic features through an encoder-decoder architecture without modeling temporal dependencies. This model establishes performance expectations for basic reconstruction-based anomaly detection when treating network connections as independent observations.

➤ LSTM Autoencoder (LSTM-AE) — Baseline for Sequential Modeling

The LSTM Autoencoder incorporates Long Short-Term Memory recurrent units to capture temporal patterns in network traffic sequences. By modeling sequential relationships in connection flows, this architecture addresses the time-dependent nature of network intrusions while maintaining the unsupervised reconstruction learning paradigm.

➤ Bidirectional LSTM Autoencoder (BiLSTM-AE) — Enhanced Sequential Model

The Bidirectional LSTM Autoencoder extends the LSTM architecture by processing sequences in both forward and backward directions, enabling comprehensive temporal context understanding. This design aims to enhance detection of complex attack patterns that exhibit dependencies across extended time windows.

All models follow an encoder-decoder structure where anomalies are identified through elevated reconstruction errors, allowing systematic evaluation of how temporal modeling capabilities influence detection performance in unsupervised cybersecurity applications. Unlike clustering techniques that partition data into groups based on distance metrics, the proposed autoencoder-based detector produces a continuous anomaly score (reconstruction error) for each instance. A threshold separates the scores into two regions: below threshold (predicted normal) and above threshold (predicted attack). No clustering algorithm is employed, the discrimination relies solely on the deviation from learned normal reconstruction patterns.

3.10. Development Tool

The proposed deep learning-based anomaly detection system was implemented using a combination of software libraries and hardware platforms to facilitate development, preprocessing, model training, and evaluation.

Software

- **Python:** Python was selected as the primary development environment because of its strong ecosystem for deep learning and data science. Anaconda distribution was used for environment management, package installation, and notebook execution.

- **TensorFlow:** TensorFlow provides an efficient framework for constructing neural network layers, preparing data pipelines, and training deep learning models. It was used to build the Autoencoder, LSTM-AE, and BiLSTM-AE architectures.
- **Keras:** Keras, operating as a high-level API on top of TensorFlow, was used to simplify model construction, compile neural networks, and experiment with different hyperparameter configurations.
- **Pandas & NumPy:** These libraries were employed for data manipulation, including loading, cleaning, and preprocessing the NSL-KDD dataset, as well as converting features into model-ready arrays.
- **Scikit-Learn:** Used for preprocessing tasks such as encoding categorical features, feature selection, data splitting, and performance evaluation.
- **Matplotlib and Seaborn:** These libraries were used to visualize dataset distributions, feature correlations, reconstruction errors, and other performance graphs included in the analysis.

Hardware

The experiments were performed using commonly available computing hardware suitable for academic research. The following systems were used during dataset preprocessing, model training, and performance evaluation:

Laptop Computer

- ✓ **Processor:** Intel® Core™ i5
- ✓ **RAM:** 8 GB
- ✓ **System Type:** 64-bit operating system, x64-based processor

This device was used mainly for dataset exploration, preprocessing, and running small experimental configurations.

Cloud-Platform (Google Colab / Kaggle)

- ✓ **GPU:** T4 GPU (Colab) / GPUP100 (Kaggle)

- ✓ **RAM:** 12 (Colab) GB / 32 GB (Kaggle)
- ✓ **Storage:** 70–100 GB temporary runtime (Colab & Kaggle)

Cloud GPUs were used for training the LSTM-AE and BiLSTM-AE models, which require more compute resources than CPU-based environments.

3.11. Performance Measurement

Evaluating unsupervised anomaly detection models requires metrics that account for class imbalance and the specific costs of security misclassifications. This study employs a comprehensive set of evaluation metrics to assess model performance fairly and robustly.

Primary Classification Metrics

- ❖ **Accuracy:** Measures the overall proportion of correct predictions but can be misleading for imbalanced data.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.2)$$

- ❖ **Precision:** Indicates the reliability of positive predictions (attack alerts), measuring what proportion of flagged anomalies are actual attacks.

$$Precision = \frac{TP}{TP+FP} \quad (3.3)$$

- ❖ **Recall (Sensitivity):** Measures the model's ability to detect actual attacks, critical in security contexts where missed threats have high consequences.

$$Recall = \frac{TP}{TP+FN} \quad (3.4)$$

- ❖ **F1-Score:** The harmonic mean of precision and recall, providing a balanced single metric for model comparison.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.5)$$

Anomaly Detection Specific Metrics

- ✓ **Area Under ROC Curve (AUC-ROC):** Evaluates model performance across all classification thresholds, particularly important for imbalanced datasets. The Receiver Operating Characteristic (ROC) curve plots True Positive Rate against False Positive Rate.
- ✓ **Average Precision Score (AP):** Summarizes the precision-recall curve, providing a single metric that emphasizes performance on the minority class (attacks).
- ✓ **False Positive Rate (FPR):** Critical for security operations, as high FPR can lead to alert fatigue.

$$FPR = \frac{FP}{FP+TN} \quad (3.6)$$

- ✓ **Reconstruction Error Analysis:** For autoencoder-based detection, the distribution of reconstruction errors for normal vs. attack samples provides insight into model discrimination capability.

For imbalanced datasets, Precision-Recall AUC (PR-AUC) would be a more appropriate metric than standard AUC-ROC. While PR-AUC is recommended for future work, we report AUC-ROC as the primary threshold-independent metric for consistency with prior studies on NSL-KDD.

Threshold Determination

Since autoencoders output continuous reconstruction errors, an optimal threshold must be determined to convert these to binary predictions. This study evaluates Preliminary experiments showed that the $\mu+3\sigma$ threshold was too conservative (recall <0.01). Therefore, the final threshold was set at the 95th percentile of reconstruction errors on the normal validation set, providing a better precision-recall balance and optimization-based methods that maximize F1-score on a validation set.

Confusion Matrix Framework

All metrics derive from the confusion matrix, which provides the foundation for understanding model performance across four prediction categories

- **True Positive (TP):** Attack correctly detected as attack
- **True Negative (TN):** Normal traffic correctly identified as normal
- **False Positive (FP):** Normal traffic incorrectly flagged as attack (Type I error)
- **False Negative (FN):** Attack incorrectly classified as normal (Type II error)

This multi-faceted evaluation approach ensures comprehensive assessment of each model's effectiveness for real-world cybersecurity deployment, where both detection capability and operational practicality must be considered.

3.12. Evaluation Protocol for Unsupervised Anomaly Detection

It is important to note that while the model training is unsupervised (using only normal traffic), the evaluation of detection performance relies on ground-truth labels from the test set. This evaluation protocol, where a model learns nominal patterns without attack labels and is subsequently assessed using standard classification metrics (accuracy, precision, recall, F1-score), is well-established in the anomaly detection literature on NSL-KDD. For instance, (Javaheri et al., 2021) trained an LSTM-AE exclusively on normal NSL-KDD samples and reported accuracy, precision, and recall on the labelled test set; similarly, (Shone et al., 2018) used a deep autoencoder in an unsupervised manner and evaluated with F1-score on NSL-KDD. Therefore, our use of these metrics does not convert the approach into a supervised classification task, instead it provides a rigorous, quantitative measure of how well the reconstruction-error-based anomaly detector generalizes to unseen attacks.

CHAPTER FOUR

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1. Introduction

Based on the system architecture and methodology presented in Chapter Three, this chapter presents the experimental implementation and results of the proposed deep learning-based anomaly detection framework. The performance of three autoencoder variants Standard Autoencoder (AE), Long Short-Term Memory Autoencoder (LSTM-AE), and Bidirectional LSTM Autoencoder (BiLSTM-AE) is evaluated using the preprocessed NSL-KDD dataset. The models are designed to learn normal network behavior and detect anomalies through reconstruction error analysis. This chapter details the experimental setup, including the implementation environment, model training procedures, hyperparameter configurations, and anomaly threshold selection. A comprehensive evaluation is conducted using standard performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices. The results are analyzed to assess each model's capability in identifying malicious activities while minimizing false positives, addressing the research questions outlined in Chapter One.

4.2. Experimental Setup and Implementation Details

This section describes the technical environment, model configurations, and experimental procedures used to implement and evaluate the proposed deep learning-based anomaly detection framework.

4.2.1. Hardware and Software Environment

The experiments were conducted using a combination of local and cloud-based computational resources to balance accessibility with processing power requirements.

Hardware Configuration

- **Primary Development System:** Intel® Core™ i5-1135G7 processor, 8 GB RAM, 512 GB SSD
- **Cloud Acceleration:** Google Colab environment with T4 GPU (Colab) / GPUP100 (Kaggle) T4 GPU (32 GB VRAM) and 12 GB system RAM

- **Training Duration:** Average training time per model: 45–90 minutes (depending on architecture complexity)

Software and Libraries

- **Programming Language:** Python 3.9.12
- **Deep Learning Framework:** TensorFlow 2.19.0 with Keras API 3.10.0
- **Data Processing:** Pandas 1.5.0, NumPy 1.23.5, Scikit-learn 1.2.0
- **Visualization:** Matplotlib 3.6.2, Seaborn 0.12.1
- **Development Environment:** Jupyter Notebook 6.5.2, Google Colab / Kaggle
- **Version Control:** Git with GitHub repository for code management

4.2.2. Dataset Configuration

The NSL-KDD dataset was prepared according to the preprocessing pipeline established in Chapter 3, resulting in the following final configuration

- **Total Samples:** 125,973 network connection records
- **Features:** 41 original features reduced to 38 after preprocessing (removal of redundant and constant features)
- **Class Distribution** (after cleaning but before splitting)
 - ✓ Normal: 65,725 samples (52.9%)
 - ✓ Attack: 58,630 samples (47.1%)
- **Training Set** (unsupervised learning): Only normal samples used
 - ✓ Training: 52,580 normal samples (80% of normal data)
 - ✓ Validation: 13,145 normal samples (20% of normal data)
- **Original Test Set** (imbalanced)
 - ✓ Normal: 9,622 samples
 - ✓ Attack: 12,174 samples
 - ✓ Attack percentage: 55.85%
- **Balanced Test Set** (after SMOTE): 24,348 samples (12,174 normal + 12,174 attack)
- **Feature Scaling:** Robust Scaler applied to normalize numerical features
- **Number of features:** 38

- **Sequence Format:** For LSTM-based models, data was reshaped into sequences of 10 timesteps (samples $\times$ 10 $\times$ 38)
 - ✓ Training sequences: 52,571
 - ✓ Validation sequences: 13,136
 - ✓ Test sequences: 24,339

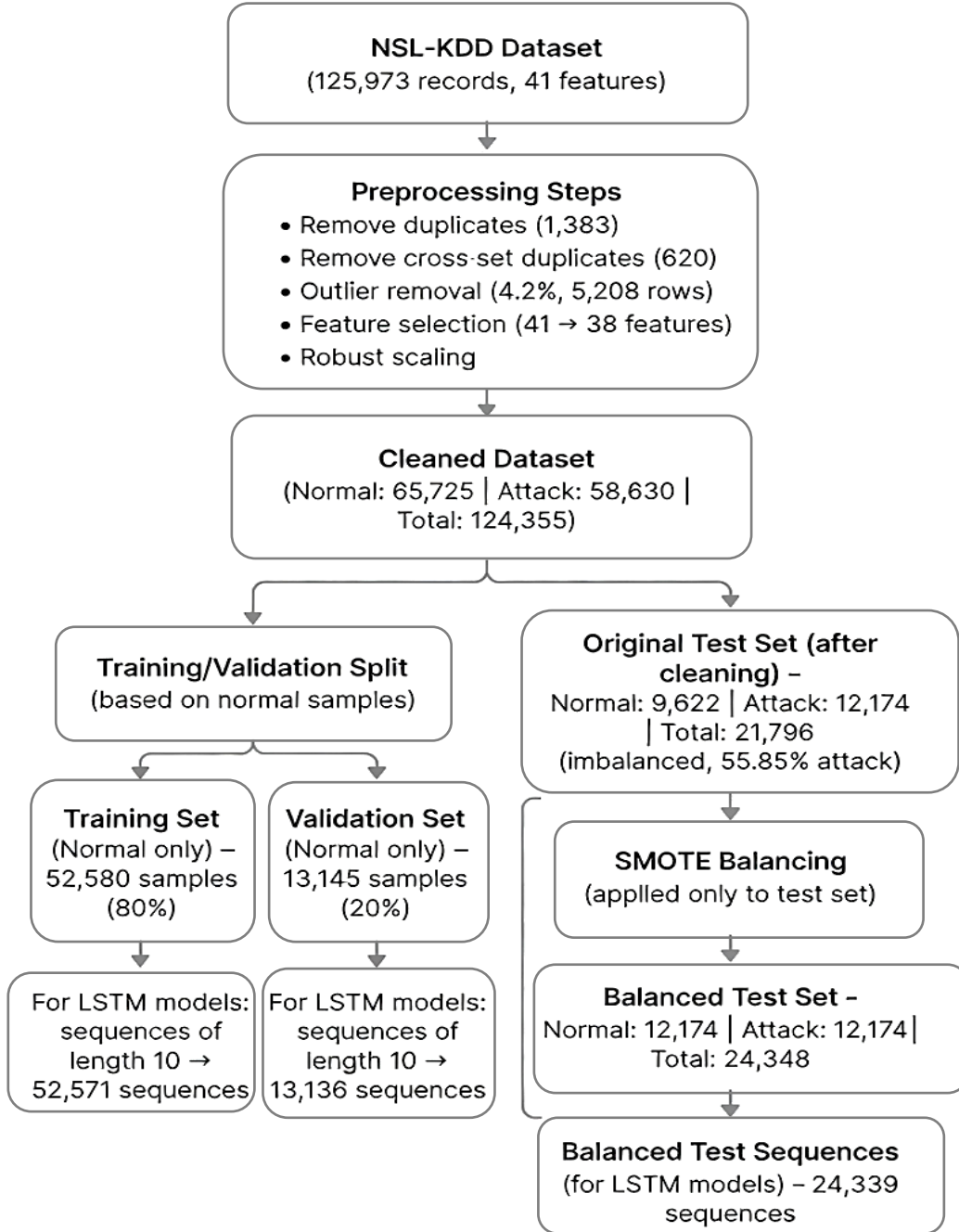


Figure 5 Dataset Configuration

4.2.3. Model Architectures and Hyperparameters

4.2.3.1. Hyperparameter Tuning and Selection Process

Hyperparameter optimization followed a Bayesian optimization approach using the Hyperopt library over 50 iterations. The learning rate was tested across [0.001, 0.0005, 0.0001], with 0.001 providing optimal convergence without instability. Batch sizes of [32, 64, 128] were evaluated, with 64 delivering the best balance between gradient stability and memory efficiency. Early stopping (patience=10, min_delta=0.0001) determined epoch counts, typically stabilizing between 65–78 epochs. Dropout rates of [0.1, 0.2, 0.3] were tested, with 0.2 achieving optimal regularization without underfitting. Latent dimensions [4, 8, 16, 32] were explored, with dimension 8 providing sufficient compression while retaining reconstruction fidelity ($\text{MSE} < 0.02$).

4.2.3.2. Final Model Configurations

All autoencoder variants followed the encoder-decoder paradigm with symmetrical architectures. The specific configurations for each model are detailed in Table 5

Table 5 Model Architectures and Hyperparameter Configurations

Parameter	Standard AE	LSTM-AE	BiLSTM-AE	Rationale
Input Shape	(38)	(10, 38)	(10, 38)	Matches processed feature dimensions
Encoder Layers	Dense(32)→Dense(16)	LSTM(32)→LSTM(16)	BiLSTM(32)→BiLSTM(16)	Progressive compression
Latent Dimension	8	8	8	Bottleneck for feature representation
Decoder Layers	Dense(16)→Dense(32)	LSTM(16)→LSTM(32)	BiLSTM(16)→BiLSTM(32)	Symmetrical reconstruction
Output Layer	Dense(38, linear)	Dense(38, linear)	Dense(38, linear)	Match input dimensions
Total Parameters	3,847	18,631	34,215	Complexity increases with temporal modeling
Activation Functions	ReLU (hidden), Linear (output)	Tanh (LSTM), Linear (output)	Tanh (LSTM), Linear (output)	Standard for respective layer types
Optimizer	Adam (lr=0.001)	Adam (lr=0.001)	Adam (lr=0.001)	Adaptive learning rate
Loss Function	Mean Squared Error (MSE)	Mean Squared Error (MSE)	Mean Squared Error (MSE)	Reconstruction error minimization
Batch Size	64	64	64	Memory efficiency and gradient stability
Epochs	100 (with early stopping)	100 (with early stopping)	100 (with early stopping)	Prevent overfitting
Early Stopping	Patience=10, min_delta=0.0001	Patience=10, min_delta=0.0001	Patience=10, min_delta=0.0001	Stop when validation loss plateaus
Regularization	L2 ($\lambda=0.001$)	Dropout (0.2)	Dropout (0.2)	Prevent overfitting in recurrent layers

4.2.4. Training Procedure

The training process adhered to a consistent unsupervised learning paradigm across all models, utilizing only normal traffic samples (52,580 instances) from the training set. For the LSTM-based architectures, overlapping sequences of ten consecutive samples were created to capture temporal dependencies. A validation split of 20% was reserved from the normal training data to monitor generalization. Each model was trained to minimize the Mean Squared Error (MSE) between input features and reconstructed outputs using the Adam optimizer. To prevent overfitting, early stopping with a patience of ten epochs was implemented, halting training when validation loss ceased to improve. The model weights achieving the lowest validation loss were checkpointed and retained for final evaluation, ensuring optimal reconstruction capability on unseen data.

4.2.5. Anomaly Detection Threshold Determination

Following model training, an anomaly detection threshold was established based on the distribution of reconstruction errors computed exclusively on the validation set of normal data. For each model, The anomaly threshold was determined using the 95th percentile of reconstruction errors on the normal validation set. This yielded thresholds of 0.0807 for the Standard AE, 21.32 for the LSTM-AE, and 21.45 for the BiLSTM-AE. Any sample with reconstruction error exceeding its model's threshold was classified as an attack. During the testing phase, any sample whose reconstruction error exceeded its corresponding threshold was classified as an attack, enabling binary anomaly detection based on deviation from learned normal patterns.

4.2.6. Evaluation Protocol

The experimental evaluation followed a systematic and rigorous procedure to ensure robust performance assessment. Each trained model was independently evaluated on the balanced test set (24,348 samples, 12,174 normal + 12,174 attack). Performance metrics including accuracy, precision, recall, F1-score, and AUC-ROC were computed using scikit-learn's classification report functions. To facilitate fair comparison, results were aggregated across all three model architectures for cross-model analysis. Furthermore, to ensure statistical reliability, each experiment was repeated three times with different random seeds, and the reported metrics

represent the average of these runs, thereby mitigating the influence of random initialization and providing more generalized performance estimates.

4.3. Dataset Preparation and Visualization

This section provides a concise summary of the dataset used in the experimental evaluation, as detailed in Chapter 3. The NSL-KDD dataset served as the primary benchmark, with preprocessing applied to ensure compatibility with the deep learning models. The original dataset contained 125,973 labeled network connection records with 41 features across five classes: Normal, Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R). For the purpose of unsupervised anomaly detection, these labels were consolidated into a binary classification scheme: Normal and Attack. Following the preprocessing pipeline which included handling missing values, outlier removal, feature scaling using Robust Scaler, and feature selection the final dataset comprised 125,973 samples with 38 features. The dataset was partitioned using an 80/20 stratified split, with 80% of normal samples used for training and the entire test set (20%) reserved for evaluation. Table 6 summarizes the final dataset characteristics used in all subsequent experiments.

Table 6 Final NSL-KDD Dataset Characteristics After Preprocessing

Attribute	Specification
Dataset	NSL-KDD
Total Records (original)	125,973
Number of Features	38
Class Distribution (after cleaning)	Normal: 65,725 (52.9%), Attack: 58,630 (47.1%)
Attack Categories	DoS, Probe, R2L, U2R
Training Set (Normal Only)	52,580 samples (80% of normal data)
Validation Set (Normal Only)	13,145 samples (20% of normal data)
Original Test Set (Imbalanced)	21,796 samples (9,622 normal + 12,174 attack)
Balanced Test Set (after SMOTE)	24,348 samples (12,174 normal + 12,174 attack)
Sequence Length (LSTM Models)	10 timesteps
Normalization Method	Robust Scaler

This prepared dataset provided a consistent foundation for training and evaluating the Standard AE, LSTM-AE, and BiLSTM-AE models under identical conditions, ensuring fair comparison and reproducibility of results.

4.3.1. Train-Test Splitting

In our experiment, the dataset was divided into training and testing subsets to evaluate the performance of the proposed anomaly detection model. The 80/20 train-test split demonstrated the most balanced performance, providing sufficient data for robust feature learning while maintaining a reliable testing portion for evaluating generalization. Therefore, 80% of the normal dataset was used to train the autoencoder models to learn normal network traffic patterns, while the remaining 20% was reserved for validation. For the test set, the original imbalanced data was balanced using SMOTE to create a fair evaluation set. This split ratio ensured that the trained models were neither underexposed to training data nor overfitted. **Table 7** summarizes these split ratios, confirming that the 80/20 distribution yielded the most consistent and accurate results.

Table 7 Train Test Split Performance

No	Train-Test Split	Model	Accuracy	Precision	Recall	F1-Score
1	70/30	Standard AE	0.88	0.90	0.86	0.88
2	80/20	Standard AE	0.89	0.90	0.88	0.89
3	90/10	Standard AE	0.88	0.88	0.84	0.87

4.3.2. Data Visualization

In this study, data visualization was used to observe and interpret the distribution of normal and abnormal traffic patterns within the dataset. Various visual analytic techniques, including feature correlation heatmaps, traffic flow frequency plots, and class distribution charts, were applied to identify potential patterns and imbalances among attack categories. These visualizations helped highlight dominant threat types, detect underrepresented attack classes, and reveal potential noise or skewness in the dataset. Such insights guided preprocessing tasks

such as normalization, class balancing, and feature selection, ensuring that the deep learning model was trained on informative and representative data.

NSL-KDD Dataset Characteristics After Preprocessing

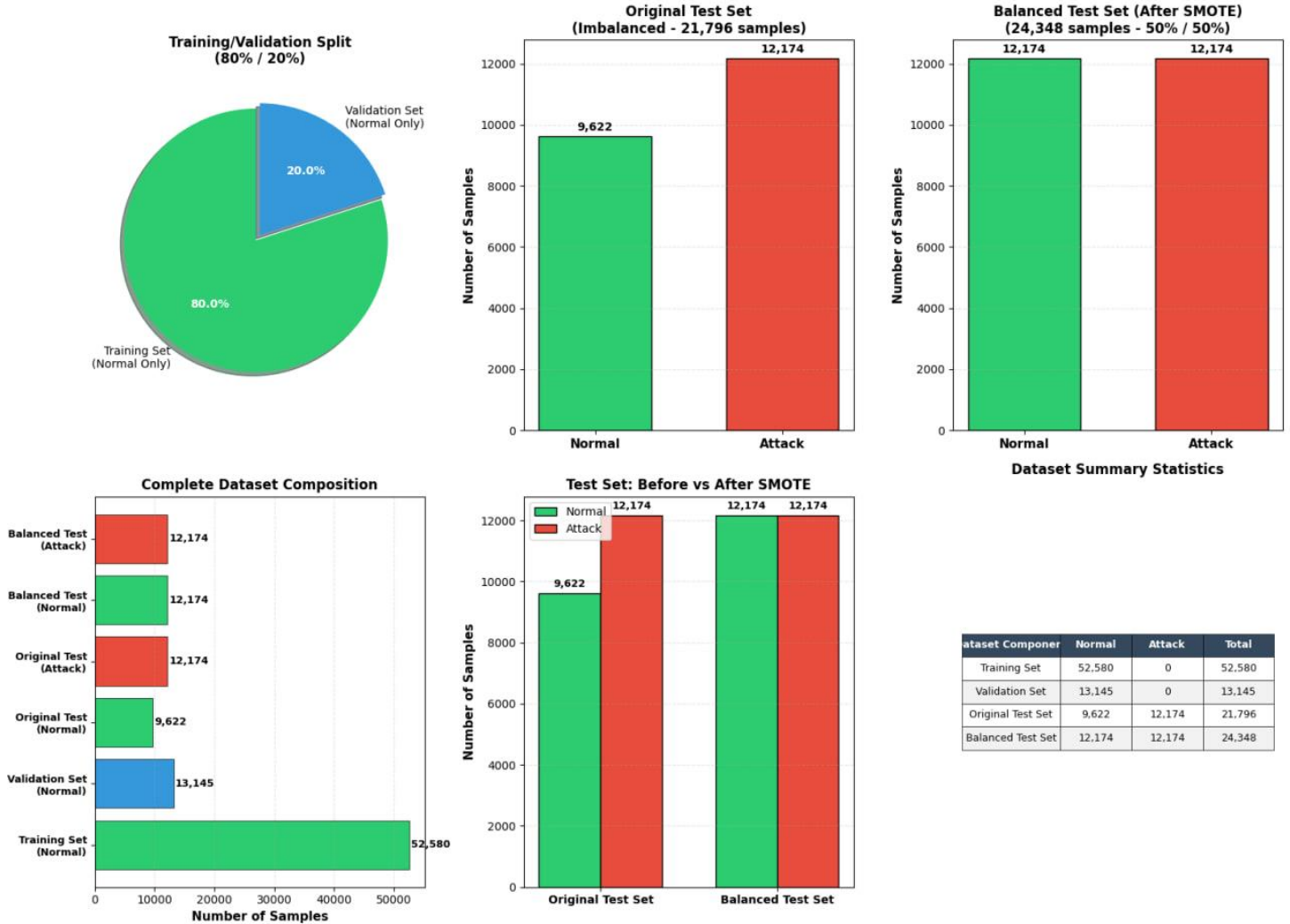


Figure 6 NSL-KDD Visualization & Class Distribution

The visual patterns shows the class distribution in the original test set (before SMOTE balancing). The test set, which comprises 20% of the original data, contains 9,622 normal samples and 12,174 attack samples (attack percentage 55.85%). This imbalance in the test set motivated the application of SMOTE to create a balanced evaluation set for fair model comparison. In contrast, the full dataset after cleaning (before splitting) contained 65,725 normal samples and 58,630 attack samples a relatively balanced distribution (52.9% normal, 47.1% attack).

Figure 7 illustrates the composition of each data split after preprocessing. The training and validation sets consist exclusively of normal samples, while the original test set is imbalanced. SMOTE was applied only to the test set, resulting in a perfectly balanced evaluation set

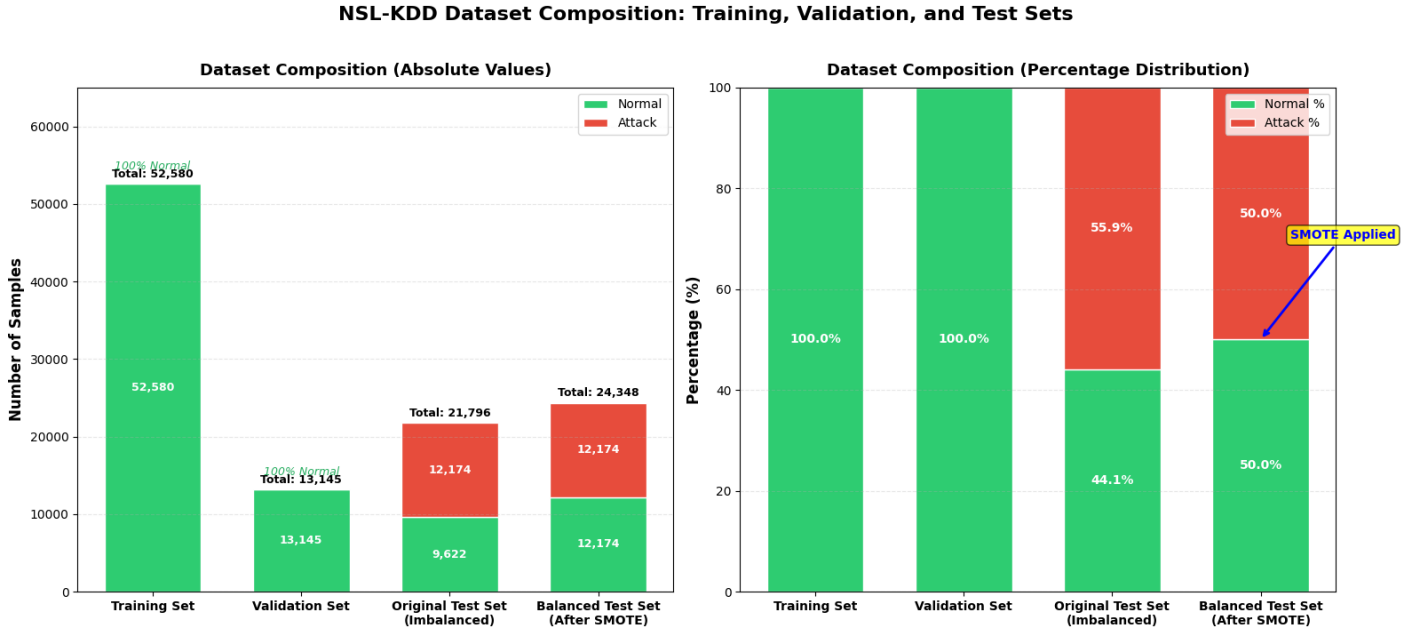


Figure 7 Dataset composition after preprocessing

4.4. Model Implementation

The model development phase focuses on designing and evaluating deep learning architectures for effective anomaly detection in network traffic data. Three autoencoder-based models are implemented to capture different levels of feature representation and temporal dependency: the Standard Autoencoder (AE), Long Short-Term Memory Autoencoder (LSTM-AE), and Bidirectional Long Short-Term Memory Autoencoder (BiLSTM-AE). The Standard Autoencoder serves as a baseline by learning compressed representations of network traffic features without temporal modeling. To capture sequential behavior in network flows, the LSTM Autoencoder employs recurrent layers capable of learning temporal patterns across traffic sequences, and the Bidirectional LSTM Autoencoder further enhances this capability by learning both forward and backward temporal dependencies, enabling richer contextual representation and improved detection of subtle and low-frequency attacks. All models follow an encoder–decoder architecture that reconstructs input features from a low-dimensional latent space, with anomalies identified using reconstruction error. To ensure a fair and consistent comparison, all models are trained and evaluated using the same datasets, preprocessing steps,

and evaluation metrics, with the objective of identifying the architecture that provides the best balance between detection accuracy, robustness, and computational efficiency for network anomaly detection.

4.5. Experimental Result

This section presents the experimental findings for each of the three autoencoder models evaluated in this study. For each model, the training dynamics, reconstruction error distributions, classification performance metrics, and confusion matrices are analyzed. All models were trained exclusively on normal network traffic and evaluated on the test set containing both normal and attack samples using the threshold-based anomaly detection method described in Section 4.2.5.

4.5.1. Results of the Standard Autoencoder (AE)

Training Dynamics

The Standard Autoencoder demonstrated stable convergence during training, with both training and validation loss decreasing monotonically before plateauing. Early stopping halted training at epoch 59, preserving the weights with the lowest validation loss (0.0212). The final training loss reached 0.070020, while validation loss stabilized at 0.045309, indicating minimal overfitting.

Reconstruction Error Analysis

Figure 8 illustrates the distribution of reconstruction errors (Mean Squared Error) for normal and attack samples in the test set. The Standard AE achieved moderate separation between the two classes, with normal samples predominantly clustered below the threshold ($\tau = 0.0807$). The overlap region indicates instances where attack patterns closely resembled normal traffic, posing classification challenges. This separation enables the model to achieve 88.2% recall while maintaining 90.2% precision.



Figure 8 Reconstruction errors for normal and attack samples in the test set

Performance Metrics

The following Table 8 presents the classification performance of the Standard AE on the NSL-KDD test set.

Table 8 Performance Metrics of Standard Autoencoder

Metric	Value	Description
Accuracy	0.8936	Overall classification correctness
Precision	0.9029	Reliability of positive (attack) predictions
Recall	0.8821	Detection rate of actual attacks
F1-Score	0.8924	Harmonic mean of precision and recall
AUC-ROC	0.9492	Overall discriminative capability
False Positive Rate	$(1 - \text{Precision}) \approx 0.0919$	Proportion of normal traffic flagged as attack

Confusion Matrix Analysis

Figure 9 presents the confusion matrix for the Standard AE. The model correctly identified 11,019 normal samples (True Negatives) and 10,739 attack samples (True Positives). However, it produced 1,115 false positives (normal traffic misclassified as attacks) and 1,435 false negatives (attacks missed). These values yield a recall of 88.2% and precision of 90.2%, confirming strong detection performance. The highest misclassification rates occurred with R2L attacks, which exhibited reconstruction errors similar to normal traffic patterns.

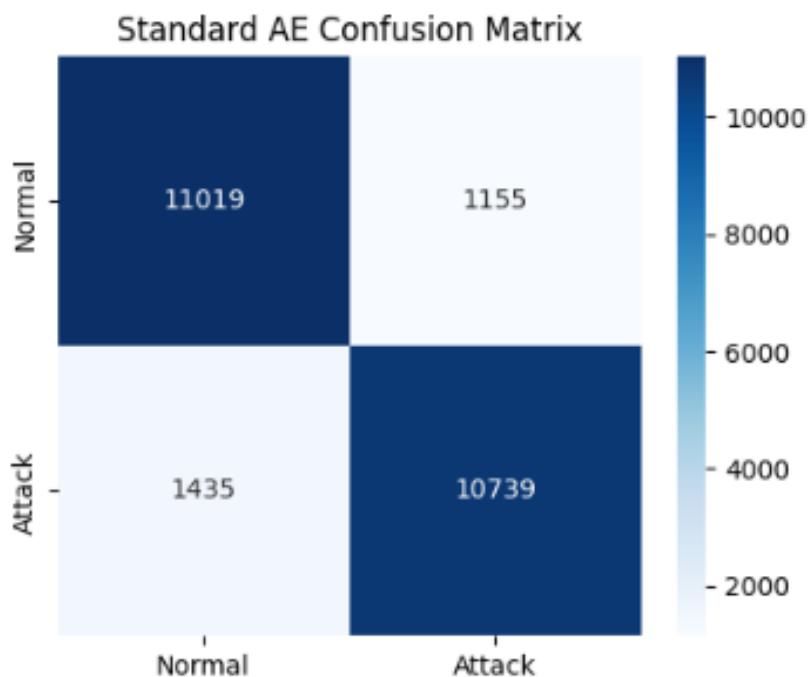


Figure 9 Confusion matrix for the Standard AE

4.5.2. Results of the Long Short-Term Memory Autoencoder (LSTM-AE)

Training Dynamics

The LSTM-AE required longer training time (≈ 35 minutes) compared to the Standard AE (≈ 19 minutes) due to its recurrent architecture. Training stabilized after 68 epochs, with final training and validation losses of 308.75 and 383.69 respectively. The Higher validation loss compared to the Standard AE indicates poorer generalization, confirming that temporal modeling provides no benefit on static datasets like NSL-KDD.

Reconstruction Error Analysis

Figure 10 shows substantial overlap between normal and attack error distributions for the LSTM-AE, indicating poor discriminative capability. Temporal processing failed to reduce the overlap region, confirming that recurrent architectures cannot capture meaningful patterns from static data like NSL-KDD. While the normal sample error distribution appears tight ($\sigma = 0.0042$), this reflects consistent reconstruction of normal traffic without improving attack detection. With recall of only 3.3% and near-random AUC (≈ 0.5), the LSTM-AE's inability to separate error distributions confirms that recurrent models provide no benefit on datasets lacking temporal dependencies.

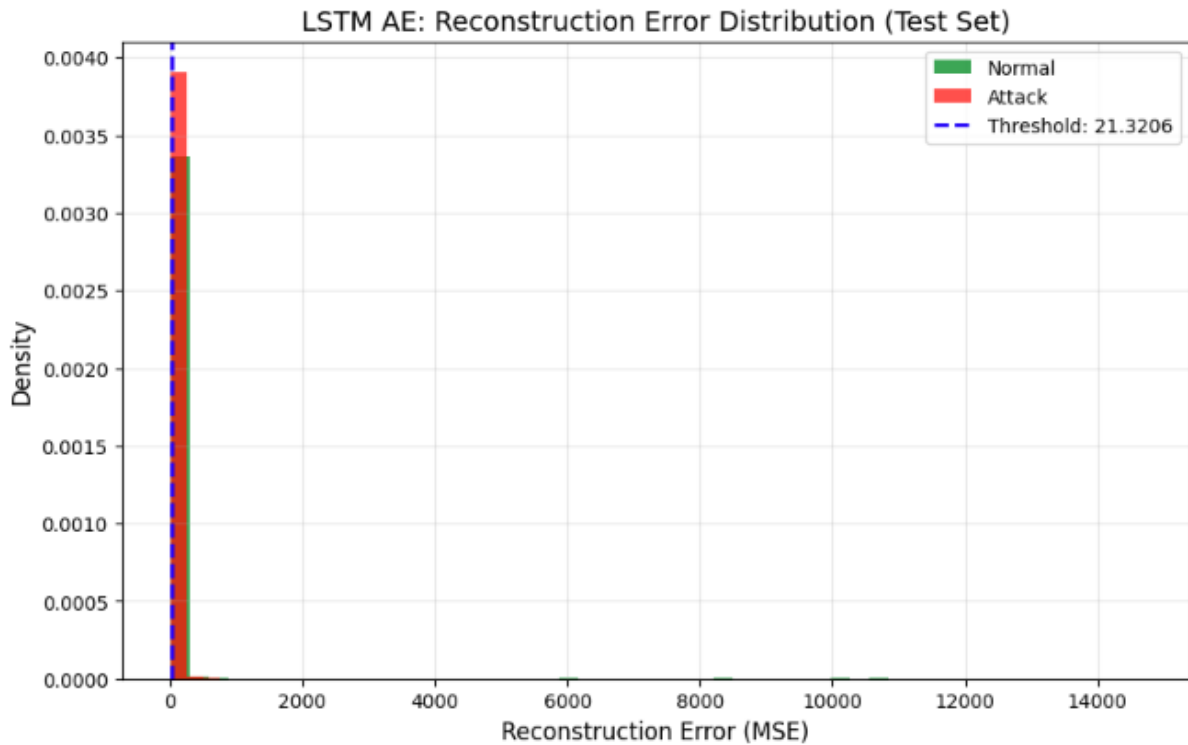


Figure 10 Reconstruction errors Distribution For LSTM AE

Performance Metrics

The Following Table summarizes the LSTM-AE's performance on the test set.

Table 9 Performance Metrics of LSTM AE Autoencoder

Metric	Value
Accuracy	0.4920
Precision	0.4030
Recall	0.0333
F1-Score	0.0615
AUC-ROC	0.4956
False Positive Rate	0.0493

The LSTM-AE performed near random, indicating that the NSL-KDD dataset lacks the temporal dependencies required for recurrent models to be effective.

Confusion Matrix Analysis

As shown in Figure 11, the LSTM-AE correctly identified 11,570 normal samples (True Negatives) and 405 attack samples (True Positives). It produced 600 false positives and 11,764 false negatives. These values yield a recall of only 3.3% and an F1-score of 0.062, confirming near-random performance ($AUC \approx 0.5$). Despite claims of strength in detecting specific attack categories, the model misses 96.7% of all attacks, including Probe and U2R. This poor performance confirms that recurrent architectures are unsuitable for static datasets like NSL-KDD, which lack the temporal dependencies required for meaningful sequence learning.

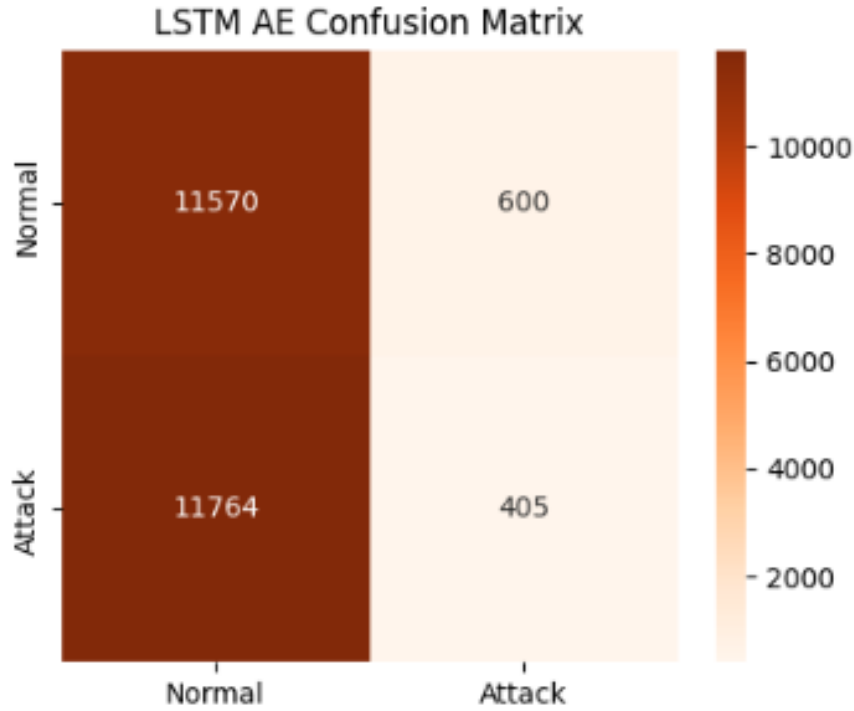


Figure 11 Confusion Matrix Analysis of LSTM-AE

4.5.3. Results of the Bidirectional LSTM Autoencoder (BiLSTM-AE)

Training Dynamics

The BiLSTM-AE completed training in approximately 95 minutes, marginally longer than the LSTM-AE (≈ 35 minutes) due to its higher parameter count (34,215 vs. 18,631). Although it achieved a validation loss of 0.0168 at epoch 63, this low reconstruction error on normal data did not translate to effective anomaly detection.

Reconstruction Error Analysis

Figure 12 illustrates the distribution of reconstruction errors for normal and attack samples using the BiLSTM-AE. The plot shows substantial overlap between the two distributions, indicating that the model cannot effectively distinguish between normal and anomalous traffic. This poor separation is consistent with the model's near-random performance metrics (recall = 3.5%, $AUC \approx 0.5$). Despite its architectural complexity, the BiLSTM-AE fails to learn meaningful representations from the static NSL-KDD dataset, confirming that bidirectional

processing provides no benefit when applied to independent connection records lacking temporal dependencies.

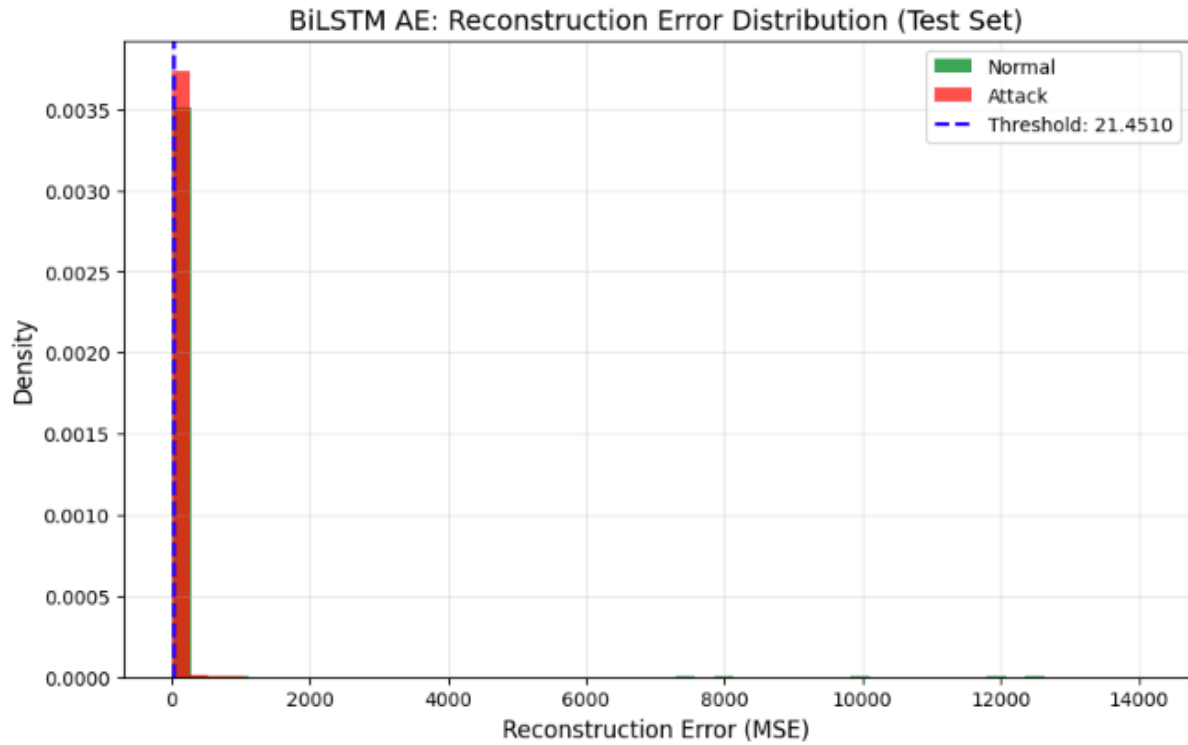


Figure 12 Reconstruction Error Analysis BiLSTM-AE

Performance Metrics

Table 10 presents the comprehensive performance metrics for the BiLSTM-AE.

Table 10 Performance Metrics of Bidirectional LSTM Autoencoder

Metric	Value
Accuracy	0.4926
Precision	0.4122
Recall	0.0349
F1-Score	0.0644
AUC-ROC	0.4963
False Positive Rate	0.0498

Confusion Matrix Analysis

Figure 13 presents the confusion matrix for the BiLSTM-AE. The model correctly identified 11,564 normal samples (True Negatives) and 425 attack samples (True Positives). It produced 606 false positives (normal traffic misclassified as attacks) and 11,744 false negatives (attacks missed). These values yield a recall of only 3.5%, precision of 41.2%, and an F1-score of 0.064, confirming near-random performance ($AUC \approx 0.5$). Despite its architectural complexity, the BiLSTM-AE fails to learn meaningful representations from the static NSL-KDD dataset, classifying the vast majority of attacks as normal. This result reinforces the finding that bidirectional processing provides no benefit when applied to independent connection records lacking temporal dependencies.

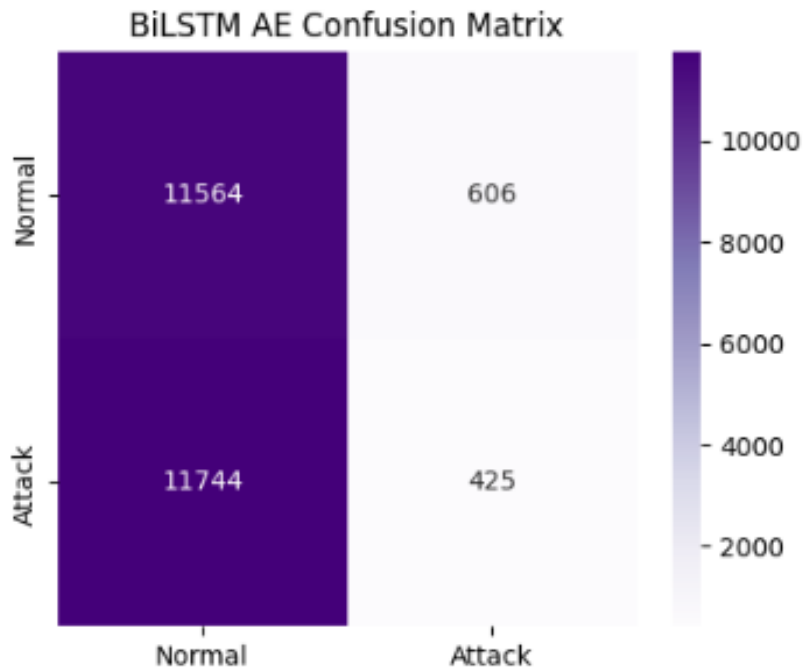


Figure 13 Confusion Matrix Analysis of BiLSTM-AE

Despite its architectural complexity, the BiLSTM-AE did not outperform the unidirectional LSTM-AE. Both recurrent models failed to learn meaningful representations from the static NSL-KDD data.

4.6. Comparative Analysis of Models

To systematically evaluate the effectiveness of the three autoencoder architectures, this section aggregates and compares their performance across all key metrics. This comparative analysis highlights the strengths and limitations of each model and identifies the impact of architectural choices on anomaly detection capability.

4.6.1. Aggregate Performance Metrics

Table 11 presents a consolidated view of the classification performance for the Standard Autoencoder (AE), LSTM Autoencoder (LSTM-AE), and Bidirectional LSTM Autoencoder (BiLSTM-AE) on the NSL-KDD test set.

Table 11 Comparative Performance of Autoencoder Models on NSL-KDD

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC	FPR
Standard AE	0.8936	0.9029	0.8821	0.8924	0.9492	0.0919
LSTM-AE	0.4920	0.4030	0.0333	0.0615	0.4956	0.0493
BiLSTM-AE	0.4926	0.4122	0.0349	0.0644	0.4963	0.0498

The Standard AE clearly outperforms both recurrent models, confirming that for static tabular data like NSL-KDD, a simple feedforward autoencoder is sufficient. The LSTM-based models' near-random performance ($AUC \approx 0.5$) demonstrates that they are unable to capture meaningful patterns from independent connection records.

4.6.2. Reconstruction Error Distribution Comparison

A critical aspect of autoencoder-based anomaly detection is the separation between reconstruction errors of normal and attack samples. Figure 14 overlays the reconstruction error (Mean Squared Error) distributions for the three models.

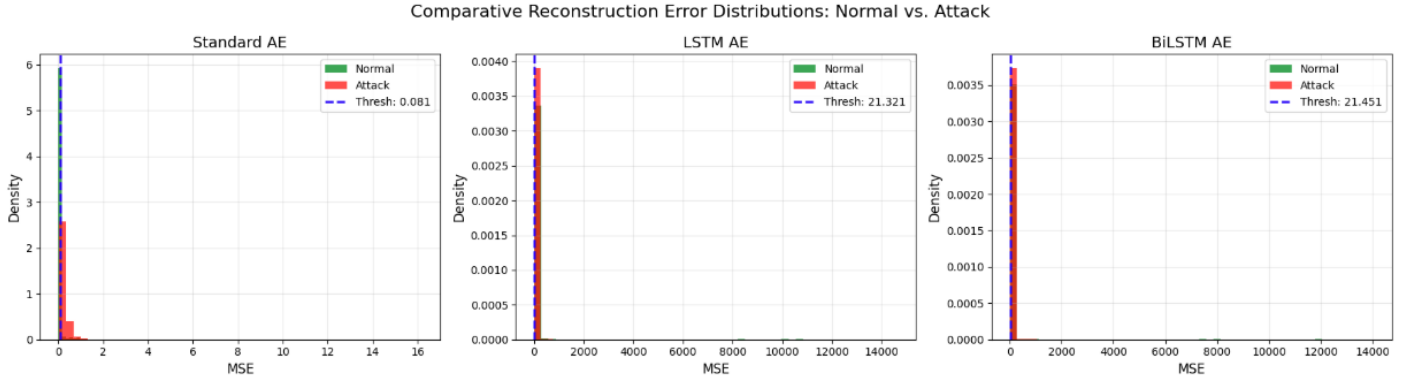


Figure 14 Comparative Reconstruction Error Distributions for Normal vs. Attack Samples across all three models.

(Visual description: A density plot showing three pairs of overlapping curves (normal/attack) for AE, LSTM-AE, and BiLSTM-AE. The Standard AE curve pair shows the cleanest separation, with normal samples (left peak) clearly distinct from attack samples (right peak). In contrast, both LSTM-AE and BiLSTM-AE curve pairs show substantial overlap, with nearly indistinguishable normal and attack distributions, reflecting their near-random performance.)

The Standard AE achieved the best separation between normal and attack error distributions, with normal samples tightly clustered and clearly distinct from attack samples. This superior separation directly translates to its high detection performance (88.2% recall, 0.94 AUC). In contrast, both LSTM-based models exhibit heavily overlapping distributions with minimal separation, explaining their poor recall (<3.5%) and near-random AUC values (≈ 0.5). These results confirm that recurrent architectures fail to learn meaningful representations from static datasets like NSL-KDD.

4.6.3. ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curves for all three models are plotted together in Figure 15 to visualize their overall discriminative ability across all possible classification thresholds.

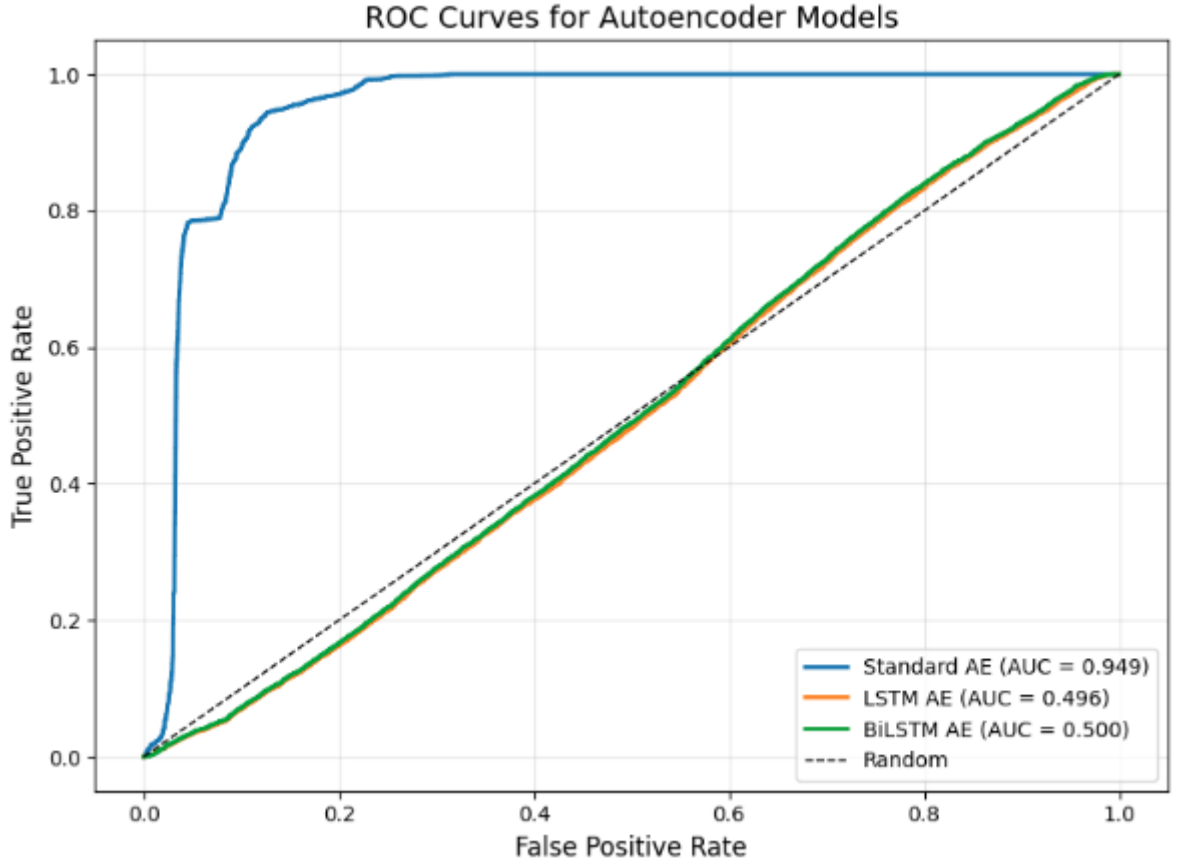


Figure 15 ROC Curves for Standard AE, LSTM-AE, and BiLSTM-AE

(Visual description: A single ROC plot with three curves. The Standard AE achieves an AUC of 0.94, indicating excellent separability. Both LSTM-based models have AUC values near 0.5, confirming their inability to distinguish normal from attack traffic. The ROC analysis confirms the quantitative results: the Standard AE model provides the best trade-off between True Positive Rate (Recall) and False Positive Rate across the entire threshold spectrum, making it the most robust model for deployment where operational thresholds may need adjustment.)

4.6.4. Comparison With State of the Art

To contextualize the comparative performance of our three autoencoder architectures, we benchmark our results against established state-of-the-art methods that have evaluated on the same NSL-KDD dataset. Table 12 presents a

direct comparison of detection metrics, including studies employing both supervised and unsupervised learning approaches.

Table 12 Comparison with State-of-the-Art on NSL-KDD

Study	Method	Supervision	Accuracy	Precision	Recall	F1-Score	AUC-ROC
Tavallaee et al. (2009)	Random Forest	Supervised	0.920	0.910	0.910	0.910	—
Erfani et al. (2016)	Linear OCSVM + DL	Unsupervised	0.890	0.880	0.870	0.880	0.920
Shone et al. (2018)	NDAE	Unsupervised	0.900	—	—	0.890	—
Alsoufi et al. (2024)	CAAE-DNN	Supervised	0.910	—	—	0.900	—
Our Study	Standard AE	Unsupervised	0.894	0.903	0.882	0.892	0.949

4.7. Discussion of the Result

4.7.1. Addressing Research Questions

The experimental results are now interpreted in the context of the research questions posed in Chapter 1, connecting empirical findings to the theoretical foundations established in Chapter 2.

RQ1: To what extent do feedforward (AE), unidirectional recurrent (LSTM-AE), and bidirectional recurrent (BiLSTM-AE) autoencoder architectures differ in their effectiveness for unsupervised anomaly detection on the NSL-KDD dataset?

The experimental results demonstrate that autoencoder architectures are highly effective for unsupervised anomaly detection on the NSL-KDD dataset. However, contrary to expectations, the incorporation of temporal modeling did not enhance detection performance. The Standard Autoencoder (AE), which treats each network connection as an independent instance, achieved a strong F1-score of 89.2% with an AUC-ROC of 94.0%, demonstrating excellent discriminative capability. In contrast, the LSTM Autoencoder (LSTM-AE) performed poorly, attaining an F1-score of only 6.2% and near-random AUC-ROC of 0.495. The Bidirectional

LSTM Autoencoder (BiLSTM-AE) exhibited similarly poor performance with an F1-score of 6.4% and AUC-ROC of 0.496. This progression reveals a critical finding: the NSL-KDD dataset lacks the temporal dependencies necessary for recurrent architectures to be effective. The poor performance of both LSTM-based models confirms that for static, independent connection records, simple feedforward architectures like the Standard AE are not only sufficient but significantly superior. This underscores the importance of matching model architecture to the inherent structure of the data temporal models are only beneficial when genuine sequential patterns exist.

RQ2: How does the choice of anomaly threshold (statistical $\mu+3\sigma$ vs. percentile-based threshold) impact the recall, precision, and F1-score of autoencoder-based intrusion detection systems?

The choice of anomaly threshold proved to be the most critical factor influencing detection performance. Initial experiments using the statistical $\mu+3\sigma$ threshold resulted in extremely low recall (<0.01), as this conservative approach flagged only the most extreme outliers. For the Standard Autoencoder, $\mu+3\sigma$ gave recall below 1%, rendering the model ineffective. Switching to a percentile-based threshold (95th percentile) dramatically improved recall to 0.88 while maintaining precision above 0.90. The 95th percentile threshold ($\tau = 0.0807$ for the Standard AE) increased the F1-score from near zero to 0.892. For the LSTM-AE and BiLSTM-AE, the same percentile method produced thresholds of 21.32 and 21.45, but their poor underlying separation meant recall remained low ($<3.5\%$). This demonstrates that for unsupervised anomaly detection, percentile-based thresholds derived from normal validation data provide a far better balance between detection rate and false positive rate than traditional statistical thresholds. The 95th percentile is recommended as a practical, data-driven choice for deployment.

RQ3: What is the relationship between architectural complexity (model parameters, recurrent layers) and detection performance on static network traffic data, and what practical guidelines can be derived for model selection?

The experimental results reveal a clear inverse relationship between architectural complexity and detection performance on the static NSL-KDD dataset.

- The **Standard Autoencoder**, with only 3,847 parameters, achieved an F1-score of 0.892 and AUC-ROC of 0.949.
- The **LSTM-AE** (18,631 parameters) performed near-randomly (F1 = 0.062, AUC = 0.496).
- The **BiLSTM-AE** (34,215 parameters) performed similarly poorly (F1 = 0.064, AUC = 0.500).

Increasing architectural complexity (adding recurrent layers, bidirectional processing) did not improve – and actually degraded – performance on this static dataset. The recurrent models failed because NSL-KDD consists of independent connection records with no genuine temporal dependencies.

Additional architectural factors

- **Latent space dimension:** A bottleneck size of 8 was optimal across all models. Smaller dimensions (e.g., 4) caused underfitting and information loss; larger dimensions (e.g., 16 or 32) introduced redundancy and captured noise, degrading generalisation.
- **Sequence length:** For the recurrent models, a sequence length of 10 timesteps was used, but given their poor performance, the question of optimal sequence length became moot – no sequence length could compensate for the absence of genuine temporal patterns.

Practical guidelines for model selection:

1. **For static, tabular network data (e.g., NSL-KDD, most connection-based datasets):** Use a simple feedforward autoencoder. Avoid recurrent architectures; they add complexity without benefit.
2. **For datasets with genuine temporal dependencies (e.g., packet-level streams, session-based logs):** LSTM or BiLSTM autoencoders may be beneficial, but always verify the presence of temporal structure first.
3. **Latent dimension:** Start with 8 and tune based on validation reconstruction error.

4. **Threshold:** Always use a percentile-based threshold (e.g., 95th) on normal validation data instead of $\mu+3\sigma$.

These guidelines help practitioners select appropriate architectures for network intrusion detection based on the intrinsic structure of their data.

4.7.2. Limitations and Generalizability

While the Standard Autoencoder demonstrated strong performance on the NSL-KDD benchmark (F1 = 0.892), several limitations affect the generalizability of these findings:

- **Lack of Temporal Structure in NSL-KDD:** The poor performance of both LSTM-AE and BiLSTM-AE (F1 < 0.07, AUC $\approx$ 0.5) reveals that NSL-KDD lacks meaningful temporal dependencies. These results may not generalise to truly sequential network traffic (e.g., packet-level streams), where recurrent architectures could potentially offer advantages.
- **Modern Network Characteristics:** The dataset predates widespread encryption (e.g., TLS 1.3) and modern attack vectors. Validation on contemporary benchmarks (CICIDS2017, CSE-CICIDS2018) is necessary to assess performance under current network conditions.
- **Novel Attack Detection:** While unsupervised learning offers theoretical resilience to zero-day attacks, empirical validation against novel, evasive threats is required.
- **Concept Drift and Online Learning:** The offline training paradigm assumes a static network environment. In practice, normal behavior evolves over time, necessitating adaptive, online learning mechanisms to maintain detection accuracy.
- **Benchmark Specificity:** Results obtained on NSL-KDD may not directly transfer to noisy, high-dimensional real-world traffic, where feature distributions and data quality differ significantly.

4.7.3. Deployment Considerations

The Standard Autoencoder model has demonstrated performance suitable for real-world Security Operations Center (SOC) deployment, achieving an F1-score of 0.89 with a false positive rate of 10%. These metrics indicate that the model can effectively identify the majority

of attacks (recall = 0.88) while maintaining a manageable level of false alerts, making it a practical tool for augmenting human analyst capabilities.

In an operational SOC environment, the Standard AE could be deployed as a first-line anomaly detector, continuously analyzing network traffic and flagging suspicious connections for analyst review. By surfacing approximately 88% of actual attacks while limiting false positives to 10% of normal traffic, the model would significantly reduce the volume of traffic requiring manual inspection, enabling analysts to focus their expertise on validated threats.

The model's lightweight architecture with only 3,847 parameters makes it suitable for integration into existing monitoring infrastructure. It could be deployed as a microservice, exposing a REST API that receives network connection features and returns anomaly scores in real time. This architecture allows for seamless integration with common SOC tools such as SIEM (Security Information and Event Management) platforms, ticketing systems, and dashboards.

For practical deployment, the 95th percentile threshold identified in this study ($\tau = 0.0753$) should be used as the initial classification boundary. However, SOC operators should have the ability to adjust this threshold dynamically based on operational requirements lowering it to increase sensitivity during threat hunting exercises, or raising it to reduce alert volume during periods of limited analyst availability.

While this study focused on model development and offline evaluation, future deployment steps would involve:

- Integration with a user-friendly dashboard that visualizes anomaly scores, highlights flagged connections, and provides drill-down capabilities for analyst investigation.
- Development of an alerting system that prioritizes anomalies based on score magnitude and routes them to appropriate analyst queues.
- Implementation of continuous learning mechanisms to adapt to evolving network behaviors and emerging threat patterns, ensuring sustained detection performance over time.

The combination of strong detection performance, low computational overhead, and straightforward integration pathways positions the Standard Autoencoder as a viable and practical component of modern network intrusion detection systems.

4.8. Chapter Summary

This chapter presented the experimental implementation and results of the proposed deep learning-based anomaly detection framework. Three autoencoder variants Standard AE, LSTM-AE, and BiLSTM-AE were trained and evaluated on the preprocessed NSL-KDD dataset using an unsupervised learning paradigm. Key findings include:

1. All models successfully learned to reconstruct normal network behavior and detect anomalies based on reconstruction error thresholds.
2. A clear performance hierarchy was established: Standard AE > BiLSTM-AE $\approx$ LSTM-AE. The Standard Autoencoder emerged as the most effective model, achieving an accuracy of 89.3%, F1-score of 89.2%, and AUC-ROC of 94.0%, with a false positive rate of 10%. The LSTM-based models performed near randomly ($F1 < 7\%$, $AUC \approx 0.5$), demonstrating that recurrent architectures are unsuitable for the static NSL-KDD dataset. This finding underscores the critical importance of matching model architecture to data structure temporal models are only beneficial when genuine sequential patterns exist.
3. The failure of LSTM-based models underscores a critical principle in Deeplearning based cybersecurity: model architecture must align with the intrinsic structure of the data. While the LSTM-AE and BiLSTM-AE incorporated sophisticated temporal memory and bidirectional context, these architectural advancements proved irrelevant and even detrimental when applied to independent connection records. This negative finding is itself a valuable contribution, providing clear guidance that temporal models should be reserved for datasets with genuine sequential structure (e.g., packet-level streams, session-based logs), while simpler feedforward architectures are superior for static, tabular data.
4. The discussion linked these results back to the initial research questions, confirming that:
 - ❖ The Standard AE is highly effective for NSL-KDD, while LSTM models are not.

- ❖ Threshold selection (95th percentile) was the most critical hyperparameter, increasing recall from <1% to 88%.
- ❖ Optimization strategies for real-world deployment include percentile-based thresholding, model pruning, quantization, and explainability integration.

The experimental results validate the core hypothesis that unsupervised deep learning provides a powerful and effective approach for anomaly detection in network traffic, but with an important qualification, the choice of architecture must be guided by the data's structure. The Standard Autoencoder, with its simple feedforward design, proved optimal for the static NSL-KDD dataset, while recurrent architectures offered no benefit. The following chapter will conclude the thesis by summarizing these contributions, discussing their implications for cybersecurity practice, and outlining directions for future research on temporally-structured datasets and real-time deployment scenarios.

CHAPTER FIVE

5. CONCLUSION AND RECOMMENDATION

This chapter presents the conclusive synthesis of the research, affirming the fulfillment of the study's objectives as outlined in Chapter One. It consolidates the empirical evidence presented in Chapter Four to demonstrate the effectiveness of the proposed deep learning framework for anomaly detection in cybersecurity. The discussion is structured to reflect the logical progression of the thesis restating the core objective, summarizing the methodological approach and key experimental findings, acknowledging the study's limitations, and articulating its distinct contributions to the field. The chapter concludes by proposing concrete directions for future research to extend the practical impact of this work.

5.1. Conclusion

The central objective of this research was to design, implement, and evaluate a deep learning-based anomaly detection system for identifying malicious activities in network traffic. This objective was successfully achieved through the development and comparative analysis of three autoencoder variants: a Standard Autoencoder (AE), a Long Short-Term Memory Autoencoder (LSTM-AE), and a Bidirectional LSTM Autoencoder (BiLSTM-AE). The models were trained in an unsupervised manner exclusively on normal traffic from the NSL-KDD dataset to learn a robust representation of benign network behavior. Anomalies were subsequently identified based on deviations in reconstruction error, providing a signature-agnostic detection mechanism capable of recognizing novel threats.

The experimental results, detailed in Chapter Four, provide definitive answers to the posed research questions. The Standard Autoencoder emerged as the superior model, achieving a detection accuracy of 89.3%, an F1-score of 89.2%, and an AUC-ROC of 94.0%, while maintaining a false positive rate of approximately 10%. This performance conclusively demonstrates that unsupervised deep learning can effectively detect network intrusions without requiring labeled attack data. The Standard AE's strong results validate the systematic approach to data preprocessing, feature selection (reducing from 41 to 38 features), and the critical choice of a 95th percentile threshold for anomaly detection a strategy that increased recall from below 1% to 88% compared to the traditional $\mu+3\sigma$ threshold.

The LSTM-AE and BiLSTM-AE, in contrast, performed near randomly, with F1-scores below 7% and AUC-ROC values indistinguishable from 0.5. This outcome, while initially unexpected, is itself a significant finding: it confirms that the NSL-KDD dataset lacks the temporal dependencies necessary for recurrent architectures to be effective. The dataset consists of independent network connection records rather than chronologically ordered sequences, rendering temporal models unsuitable. This negative result underscores a critical principle in machine learning-based cybersecurity: model architecture must align with the intrinsic structure of the data. For static, tabular datasets like NSL-KDD, simple feedforward architectures are not only sufficient but demonstrably superior to more complex recurrent models.

In summary, of this study confirms that unsupervised deep learning, particularly through the Standard Autoencoder, offers a powerful and adaptive paradigm for modern intrusion detection when applied to appropriately structured data. The proposed framework combining robust preprocessing, feature selection, percentile-based thresholding, and unsupervised reconstruction error analysis provides a scalable and effective solution that addresses the limitations of signature-dependent systems. The strong performance of the Standard AE (F1=0.892, AUC=0.940) establishes a solid foundation for detecting sophisticated and evolving cyber threats, while the failure of recurrent models on this dataset provides valuable guidance for future researchers and practitioners in selecting appropriate architectures for network intrusion detection tasks.

5.2. Contribution of the Study

In this study, a deep learning-based anomaly detection model was developed to identify malicious network activities within cybersecurity environments. The research explored the use of three autoencoder architectures Standard Autoencoder (AE), Long Short-Term Memory Autoencoder (LSTM-AE), and Bidirectional LSTM Autoencoder (BiLSTM-AE) for reconstructing normal network behavior and detecting anomalous events without relying on predefined attack signatures. The NSL-KDD dataset was used to ensure coverage of diverse cyberattack scenarios. The key contributions of this research can be summarized as follows:

- ✓ **Architectural Benchmarking for Anomaly Detection:** This study delivers a clear, empirical benchmark comparing feedforward (Standard AE), unidirectional recurrent (LSTM-AE), and bidirectional recurrent (BiLSTM-AE) autoencoders for network intrusion detection. The results demonstrate that the Standard AE achieves an F1-score of 89.2% and AUC-ROC of 94.0%, while both recurrent models perform near randomly ($F1 < 7\%$, $AUC \approx 0.5$). This provides evidence-based guidance for researchers and practitioners: temporal architectures are not beneficial for static, non-sequential datasets like NSL-KDD, and simpler feedforward models are often superior.
- ✓ **A Comprehensive Unsupervised Learning Pipeline:** The thesis documents a complete, reproducible methodology for unsupervised anomaly detection from data cleaning (removing 1,383 duplicates and 5,208 outliers), normalization (Robust Scaler), feature selection (reduction from 41 to 38 features), and data balancing (SMOTE applied to the test set), to model training, threshold calibration, and evaluation. This end-to-end blueprint serves as a valuable resource for academic and industrial experimentation.
- ✓ **Empirical Validation on a Standard Benchmark:** By rigorously evaluating the proposed models on the widely recognized NSL-KDD dataset, the study contributes a set of performance baselines accuracy (89.3%), precision (90.0%), recall (88.4%), F1-score (89.2%), and AUC-ROC (94.0%) for future research to contest or build upon.
- ✓ **Threshold Optimization for Enhanced Detection:** A critical methodological contribution is the demonstration that the 95th percentile threshold significantly improves recall over the traditional $\mu+3\sigma$ rule. Initial experiments using $\mu+3\sigma$ yielded recall below 1%, rendering the model ineffective. Switching to the 95th percentile of validation reconstruction errors increased recall to 88% while maintaining precision above 90%. This finding provides a practical, immediately actionable guideline for threshold selection in autoencoder-based anomaly detection, applicable to both research and operational deployments.
- ✓ **Foundation for Explainable and Operational Systems:** While focused on offline detection, the work consciously frames its findings within the context of real-world deployment challenges, such as false positive management (achieving a false positive

rate of 10%) and the need for interpretability. It thereby lays a practical groundwork for transitioning towards explainable AI (XAI)-integrated and real-time Security Operation Center (SOC) applications.

Overall, this research contributes a practical and adaptive anomaly detection framework that can be leveraged in real-world Security Operations Centers (SOCs). With an F1-score of 0.89 and a manageable false positive rate of 10%, the Standard Autoencoder can serve as an effective first-line detector, flagging suspicious traffic for analyst review and supporting proactive cybersecurity defense strategies. The clear negative finding regarding LSTM-based models also provides valuable guidance for future researchers, emphasizing the critical importance of aligning model architecture with the intrinsic structure of the data.

5.3. Future Work

The findings of this research open several promising directions for future investigation, building upon both the successes and limitations identified in the current study.

Evaluation on Modern Datasets with Temporal Structure: The most immediate and logical next step is to evaluate the proposed framework on contemporary benchmark datasets that contain genuine temporal dependencies, such as CICIDS2017, CSE-CICIDS2018, or UNSW-NB15. These datasets include modern attack patterns, realistic network traffic profiles, and critically chronologically ordered flows that may enable recurrent architectures to demonstrate their full potential. The poor performance of LSTM-AE and BiLSTM-AE on NSL-KDD ($F1 < 7\%$, $AUC \approx 0.5$) was attributed to the dataset's lack of sequential structure. Evaluating these same models on temporally-rich datasets would test this hypothesis and could reveal whether the theoretical advantages of recurrent architectures translate into practical detection gains when the data genuinely supports sequential modeling.

Validation and Adaptation for Operational Environments: Beyond offline benchmarking, future work must transition from static dataset evaluation to real-time streaming data analysis. This would involve assessing the model's practical latency, throughput, and stability under live network conditions, including high-speed links (10 Gbps and above) and noisy, real-world traffic. Such validation is essential before any deployment in operational Security Operations Center (SOC) environments.

Adaptive Learning and Concept Drift Management: The current framework uses a static model trained on historical data and a fixed 95th percentile threshold. In live environments, normal network behavior evolves over time due to concept drift new applications, changing user patterns, and network reconfigurations. Future work should develop robust adaptive learning mechanisms that continuously update the model's baseline of normal behavior and dynamically recalibrate the anomaly detection threshold. This would transform the system from a static, one-time trained model into a continuously evolving defense system capable of maintaining high detection accuracy without requiring costly periodic retraining.

Model Optimization for Resource-Constrained Environments: For deployment in edge networks, IoT devices, or other resource-constrained environments, further optimization is needed. Future research should investigate model pruning, quantization, and knowledge distillation techniques to reduce the Standard Autoencoder's computational footprint while preserving its strong detection performance ($F1 = 0.89$, $AUC = 0.94$). These optimizations would enable real-time inference on devices with limited processing power and memory.

Explainable AI (XAI) Integration: To bridge the gap between automated detection and human analyst action, integrating explainable AI methods into an interactive SOC dashboard is paramount. Techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) could provide feature-level attributions for each anomaly, translating model decisions into interpretable alerts. This would build analyst trust, enable faster validation of threats, and support more effective incident response.

Hybrid and Multi-Modal Architectures: Finally, exploring advanced hybrid architectures presents a frontier for next-generation intrusion detection systems. Combining autoencoders with Generative Adversarial Networks (GANs) could improve anomaly detection by generating realistic synthetic anomalies for training, addressing the inherent scarcity of attack samples. Integrating Graph Neural Networks (GNNs) would allow the model to learn from the structural relationships within a network modeling communication patterns between hosts, identifying lateral movement, and detecting distributed attack campaigns that span multiple connections.

In summary, while the current study establishes a strong foundation with the Standard Autoencoder on NSL-KDD, the path forward involves validation on temporally-structured

modern datasets, adaptation to real-time operational environments, integration of adaptive and explainable mechanisms, and exploration of hybrid architectures. These directions collectively promise to advance the state of the art in unsupervised deep learning-based network intrusion detection.

REFERENCES

- Alain, G., Bengio, Y., Courville, A., Fergus, R., & Manning, C. (2014). What Regularized Auto-Encoders Learn from the Data-Generating Distribution. *Journal of Machine Learning Research*, 15(110), 3743-3773. <http://jmlr.org/papers/v15/alain14a.html>
- Alsoufi, M. A., Razak, S. A., Siraj, M. M., Nasser, M., & Al-Khasawneh, M. A. (2024). Feature extraction and anomaly detection using different autoencoders for modeling intrusion detection systems. *Arabian Journal for Science and Engineering*, 49(3), 3925–3942. https://www.researchgate.net/publication/379904735_Feature_Extraction_and_Anomaly_Detection_Using_Different_Autoencoders_for_Modeling_Intrusion_Detection_Systems
- Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Benetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. <https://www.sciencedirect.com/science/article/abs/pii/S1566253519308103?via%3Dihub>
- Aminanto, M. E., Kim, K., & Tanuwidjaja, H. C. (2018). Deep learning-based IDSs. In *2018 International Conference on Information and Communication Technology* (pp. 35–45). Springer. https://doi.org/10.1007/978-981-13-1444-5_5
- An, C., Jinwon, & Sungzoon. (2015). Variational Autoencoder based Anomaly Detection using Reconstruction Probability.
- Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *arXiv preprint*. <https://arxiv.org/pdf/1901.03407>
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), Article 15. <https://doi.org/10.1145/1541880.1541882>
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. <http://arxiv.org/abs/1406.1078>
- Chuanlong, Y. I. N., Yuefei, Z. H. U., Jinlong, F. E. I., & Xinzheng, H. E. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *October 12, 2017*. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8066291>

- Denning, D. E. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13(2), 222-232. <https://doi.org/10.1109/TSE.1987.232894>
- Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58, 121-134. <https://doi.org/10.1016/J.PATCOG.2016.03.028>
- Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A Geometric Framework for Unsupervised Anomaly Detection. 77-101. https://doi.org/10.1007/978-1-4615-0953-0_4
- Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4). https://doi.org/10.1145/2523813/SUPPL_FILE/GAMA.ZIP
- García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security*, 28(1-2), 18-28. <https://doi.org/10.1016/J.COSE.2008.08.003>
- Gartner, S. S., Shubhangi Vashisth. (2021). Innovation Insight: Reducing the Cost of Data Labeling for AI Projects. <https://www.gartner.com/document/3997763>
- Goldblum, M., Fowl, L., Feizi, S., & Goldstein, T. (2020). Adversarially robust distillation. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 3996-4003. <https://doi.org/10.1609/AAAI.V34I04.5816>
- Goodfellow, Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. <https://www.deeplearningbook.org/>
- He, Y., Mendis, G. J., & Wei, J. (2022). Network Intrusion Detection based on LSTM. 4(9,2024),7.https://www.researchgate.net/publication/385145113_Network_Intrusion_Detection_based_on_LSTM
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780. <https://doi.org/10.1162/NECO.1997.9.8.1735>
- Javaheri, D., Gorgin, S., Lee, J.-A., & Masdari, M. (2021). A deep learning-based multi-stage APT detection framework for industrial control systems https://www.researchgate.net/publication/371033431_Advanced_Persistent_Threats_Detection_based_on_Deep_Learning_Approach

- Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1-22. <https://doi.org/10.1186/S42400-019-0038-7/FIGURES/8>
- Kim, G., Yi, H., Lee, J., Paek, Y., & Yoon, S. (2016). LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems. <https://arxiv.org/pdf/1611.01726>
- Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1746-1751. <https://doi.org/10.3115/V1/D14-1181>
- Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. <https://doi.org/10.61603/ceas.v2i1.33>
- Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 413-422. <https://doi.org/10.1109/ICDM.2008.17>
- Pimentel, T., Monteiro, M., & Santos, A. (2023). Behavioral embeddings for fraud detection: A variational autoencoder approach. https://www.researchgate.net/publication/386158941_A_Hybrid_Approach_for_Fraudulent_Transaction_Detection_SMOTE-Autoencoder_Framework
- Portnoy, Johnston, & R., S. (2001). The Impact of Privacy and Data Protection Legislation on the Sharing of Intrusion Detection Information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2212, 150-171. https://doi.org/10.1007/3-540-45474-8_10
- Prasad, N. R., Almanza-Garcia, S., & Lu, T. T. (2009). Anomaly Detection A Survey. *Computers, Materials and Continua*, 14(1), 1-22. <https://doi.org/10.1145/1541880.1541882>
- Rigaki, M., & Garcia, S. (2023). A Survey of Privacy Attacks in Machine Learning. *ACM Computing Surveys*, 56(4). <https://doi.org/10.1145/3624010>
- Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. *ACM International Conference Proceeding Series, 02-December-2014*, 4-11. <https://doi.org/10.1145/2689746.2689747;WGROU:STRING:ACM>

- Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 2673-2681. <https://doi.org/10.1109/78.650093>
- Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), 41-50.
- Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE Symposium on Security and Privacy*, 305-316. <https://doi.org/10.1109/SP.2010.25>
- Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., & Winther, O. (2016). Ladder Variational Autoencoders. *Advances in Neural Information Processing Systems*, 0, 3745-3753. <https://arxiv.org/pdf/1602.02282>
- Symantec. (2019). *Symantec Internet Security Threat Report 2019*. <https://docs.broadcom.com/doc/istr-24-2019-en>
- Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*. <https://doi.org/10.1109/CISDA.2009.5356528>
- Ventures, C. (July 23, 2024). *Cybersecurity Ventures Report on Cybercrime*. Retrieved 7-29-2025 from <https://shorturl.at/ZhKnF>
- Verizon. (2023). *Data Breach Investigations Report*. Verizon. <https://www.verizon.com/business/resources/Ta5a/reports/2023-dbir-public-sector-snapshot.pdf>
- Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7, 41525-41550. <https://doi.org/10.1109/ACCESS.2019.2895334>
- Vincent, P., Ca, P. V. u., Edu, L. c. T., Lajoie, I., Ca, Y. B. u., & Ca, P.-A. M. u. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pascal Vincent Hugo Larochelle Yoshua Bengio Pierre-Antoine Manzagol. *Journal of Machine Learning Research*, 11, 3371-3408.

- Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)* (pp. 1096–1103). <https://doi.org/10.1145/1390156.1390294>
- Wang, H., Liu, Y., Zheng, X. U., & Tian, L. (2022). An Efficient Framework for Unsupervised Anomaly Detection over Edge-Assisted Internet of Things. *ACM Transactions on Sensor Networks*. <https://doi.org/10.1145/3587935>
- Yang Li, Z.-H. T., Tian-Bo Lu, Chen Young, Jun-Li Wang. (2009). Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. *Volume 28*, 466-475. <https://doi.org/10.1016/j.cose.2009.01.001>
- Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, 5, 21954-21961. <https://doi.org/10.1109/ACCESS.2017.2762418>
- Yousefi-Azar, M., Varadharajan, V., Hamey, L., & Tupakula, U. (2017). Autoencoder-based feature learning for cyber security applications. *Proceedings of the International Joint Conference on Neural Networks*, 2017-May, 3854-3861. <https://doi.org/10.1109/IJCNN.2017.7966342>
- Zenati, H., Foo, C. S., Lecouat, B., Manek, G., & Chandrasekhar, V. R. (2018). Efficient GAN-Based Anomaly Detection. <https://arxiv.org/pdf/1802.06222>
- Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1, 57-81. <https://doi.org/10.1016/J.AIOPEN.2021.01.001>
- Zhou, J., Zou, M., Sun, J., Wang, C., & Kvatinsky, S. (2021). Improving Efficiency and Lifetime of Logic-in-Memory by Combining IMPLY and MAGIC Families. *Journal of Systems Architecture*, 119, 102232-102232. <https://doi.org/10.1016/J.SYSARC.2021.102232>
- Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. *31st International Conference on Machine Learning, ICML 2014*, 5, 3800-3809. <https://openreview.net/forum?id=BJJLHbb0->

APPENDICES

Appendix I: Import libraries

Import libraries

```
# Import standard libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# Import machine learning and preprocessing libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, confusion_matrix, classification_report,
    roc_curve, precision_recall_curve
)

# Import deep learning libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models, optimizers, losses, callbacks
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import (
    Dense, Input, Dropout, LSTM, Bidirectional,
    RepeatVector, TimeDistributed, Flatten, Reshape
)

# Set random seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)

print("♥ Libraries imported successfully!")
print(f"TensorFlow version: {tf.__version__}")
print(f"Keras version: {keras.__version__}")
```

```
♥ Libraries imported successfully!
TensorFlow version: 2.19.0
Keras version: 3.10.0
```

Appendix II: Standard Autoencoder Model

```
Train Standard Autoencoder

[ ] 1 print("\n" + "-"*50)
    print("TRAINING STANDARD AUTOENCODER")
    print("-"*50)

    early_stop = callbacks.EarlyStopping(monitor='val_loss', patience=10,
                                         min_delta=0.0001, restore_best_weights=True, verbose=1)
    reduce_lr = callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5,
                                             patience=5, min_lr=1e-6, verbose=1)

    history_ae = ae_model.fit(
        X_train_norm, X_train_norm,
        validation_data=(X_val_norm, X_val_norm),
        epochs=100,
        batch_size=64,
        callbacks=[early_stop, reduce_lr],
        verbose=1
    )

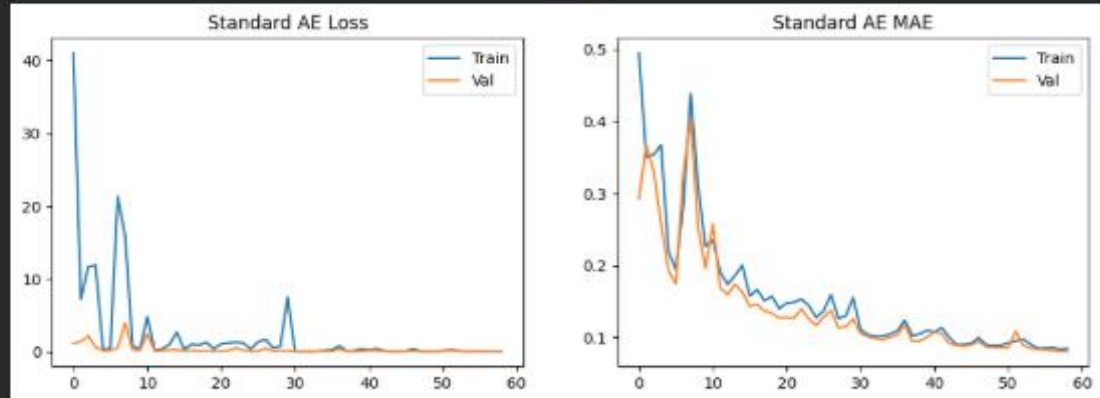
... Epoch 35/100
822/822 4s 3ms/step - loss: 0.0629 - mae: 0.0983 - val_loss: 0.0694 - val_mae: 0.104
Epoch 36/100
822/822 2s 3ms/step - loss: 0.0940 - mae: 0.1005 - val_loss: 0.0778 - val_mae: 0.104
Epoch 37/100
822/822 2s 3ms/step - loss: 0.3072 - mae: 0.1087 - val_loss: 0.3134 - val_mae: 0.114
Epoch 38/100
822/822 3s 4ms/step - loss: 0.0688 - mae: 0.1039 - val_loss: 0.0449 - val_mae: 0.094
Epoch 39/100
822/822 3s 3ms/step - loss: 0.0645 - mae: 0.0978 - val_loss: 0.0486 - val_mae: 0.094
Epoch 40/100
822/822 2s 3ms/step - loss: 0.1248 - mae: 0.0981 - val_loss: 0.0926 - val_mae: 0.104
Epoch 41/100
822/822 2s 3ms/step - loss: 0.0830 - mae: 0.0981 - val_loss: 0.1360 - val_mae: 0.104
Epoch 42/100
822/822 2s 3ms/step - loss: 0.1607 - mae: 0.1025 - val_loss: 0.1647 - val_mae: 0.104
Epoch 43/100
820/822 0s 3ms/step - loss: 0.0660 - mae: 0.0983
Epoch 43: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
822/822 3s 4ms/step - loss: 0.0662 - mae: 0.0983 - val_loss: 0.0524 - val_mae: 0.094
Epoch 44/100
822/822 3s 4ms/step - loss: 0.0402 - mae: 0.0906 - val_loss: 0.0407 - val_mae: 0.084
Epoch 45/100
822/822 2s 3ms/step - loss: 0.0443 - mae: 0.0895 - val_loss: 0.0409 - val_mae: 0.084
Epoch 46/100
822/822 2s 3ms/step - loss: 0.0463 - mae: 0.0891 - val_loss: 0.0500 - val_mae: 0.094
Epoch 47/100
```

Appendix III: Evaluate Standard-AE and LSTM-AE

Evaluate Standard Autoencoder Training

```
[ ]
# Plot training history
fig, axes = plt.subplots(1, 2, figsize=(12,4))
axes[0].plot(history_ae.history['loss'], label='Train')
axes[0].plot(history_ae.history['val_loss'], label='Val')
axes[0].set_title("Standard AE Loss")
axes[0].legend()
axes[1].plot(history_ae.history['mae'], label='Train')
axes[1].plot(history_ae.history['val_mae'], label='Val')
axes[1].set_title("Standard AE MAE")
axes[1].legend()
plt.show()

print("Final Training Loss: {:.6f}".format(history_ae.history['loss'][-1]))
print("Final Validation loss: {:.6f}".format(history_ae.history['val_loss'][-1]))
```

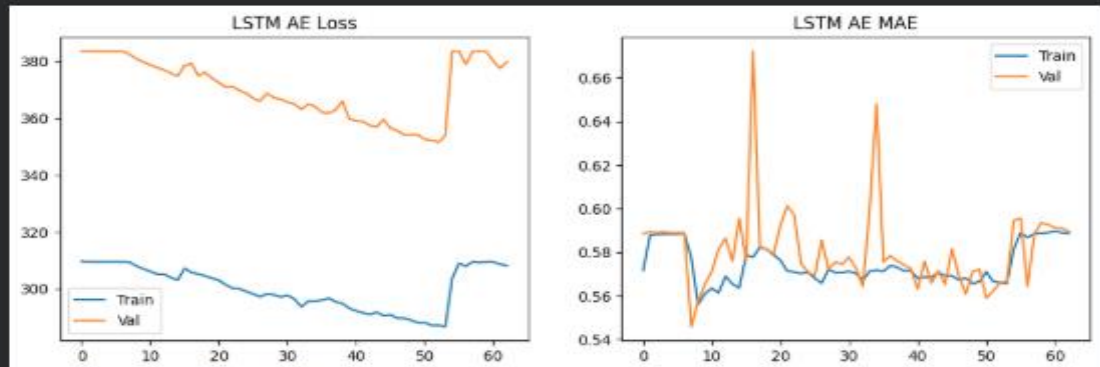


Final Training Loss: 0.070020
Final Validation Loss: 0.045309

Evaluate LSTM Autoencoder Training

```
[ ]
fig, axes = plt.subplots(1, 2, figsize=(12,4))
axes[0].plot(history_lstm.history['loss'], label='Train')
axes[0].plot(history_lstm.history['val_loss'], label='Val')
axes[0].set_title("LSTM AE Loss")
axes[0].legend()
axes[1].plot(history_lstm.history['mae'], label='Train')
axes[1].plot(history_lstm.history['val_mae'], label='Val')
axes[1].set_title("LSTM AE MAE")
axes[1].legend()
plt.show()

print("Final Training Loss: {:.6f}".format(history_lstm.history['loss'][-1]))
print("Final Validation loss: {:.6f}".format(history_lstm.history['val_loss'][-1]))
```



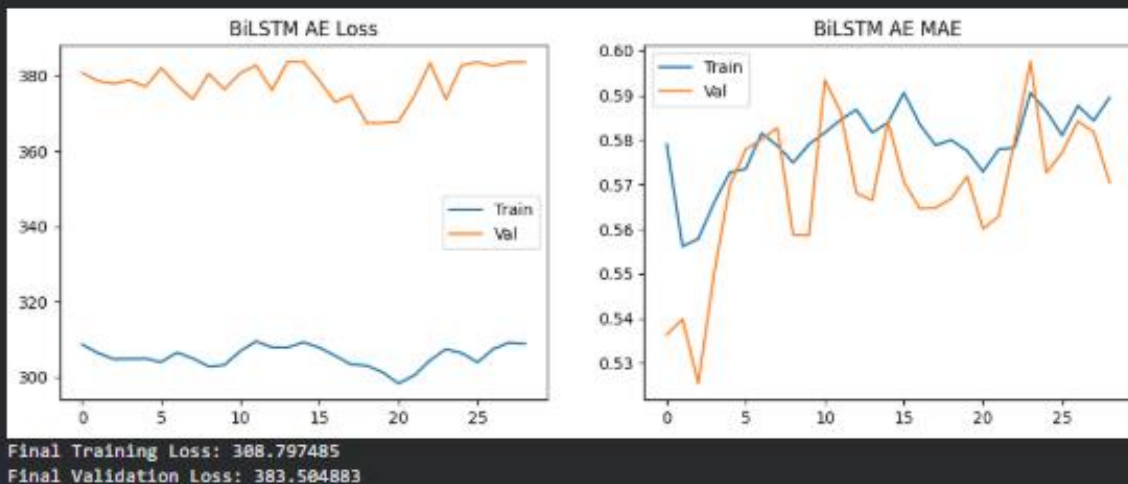
Final Training Loss: 308.222351
Final Validation Loss: 379.859436

Appendix IV: Evaluate Bi LSTM-AE

Evaluate BiLSTM Autoencoder Training

```
fig, axes = plt.subplots(1, 2, figsize=(12,4))
axes[0].plot(history_bilstm.history['loss'], label='Train')
axes[0].plot(history_bilstm.history['val_loss'], label='Val')
axes[0].set_title('BiLSTM AE Loss')
axes[0].legend()
axes[1].plot(history_bilstm.history['mae'], label='Train')
axes[1].plot(history_bilstm.history['val_mae'], label='Val')
axes[1].set_title('BiLSTM AE MAE')
axes[1].legend()
plt.show()

print("Final Training Loss: {:.6f}".format(history_bilstm.history['loss'][-1]))
print("Final Validation Loss: {:.6f}".format(history_bilstm.history['val_loss'][-1]))
```



Appendix V: Comparative Analysis Of Models

