National College of Ireland

NT-MAMBA- Network Traffic Anomaly Detection

MSc Research Project
Msc in Data Analytics

Sai Priya Unnava
Student ID: X23329173

School of Computing
National College of Ireland

Supervisor: John Kelly

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Sai Priya Unnava

Student ID: X23329173

Programme: Msc in Data Analytics **Year:** 2024-2025

Module: Msc Research Practicum

Supervisor: John Kelly

Submission Due Date: 15-09-2025

Project Title: NT-MAMBA- Network Traffic Anomaly Detection

Word Count: 8380 **Page Count:** 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sai Priya Unnava

Date: 15-09-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

NT-MAMBA- Network Traffic Anomaly Detection

Sai Priya Unnava
Student ID: X23329173

Abstract

Network traffic anomaly detection has severe issues with trade-offs between detection accuracy and computational efficiency as sophisticated cyber-attacks increase in variety and network speeds grow exponentially. Contemporary networks have millions of flows per second, and attacks vary from microsecond-level port scans to day-long infiltration attacks. Heterogeneous-natured network attacks, from volumetric DDoS to stealthy lateral movement, require detection mechanisms that can discover patterns over more than one temporal and structural dimension.

Classic approaches possess inherent trade-offs: machine learning models provide computational efficiency but sacrifice sophisticated attack behaviors, and deep learning models benefit from higher accuracy at computationally costly prices. State-of-the-art RNN and transformer-based approaches possess $O(n^2)$ complexity related to sequence length and therefore become bottlenecks to real-time deployment on high-throughput scenarios.

This research introduces NT-MAMBA, a new architecture combining multi-scale state space models (Mamba) and graph neural networks (GNN) towards efficient network anomaly detection. Three parallel Mamba scales, used in the architecture, process temporal patterns at varying resolutions, while dual GNN pathways (GraphSAGE and GAT) extract structural relationships between network entities. Temporal and spatial representations are fused through an attention-based fusion mechanism for identifying attacks occurring at varying time scales and communication patterns. Selective state space models are utilized for efficient sequence processing with preserved representational capability through graph-based structural analysis.

Analysis on CSE-CIC-IDS2018 dataset (437,498 samples, 10 attack types) discovers 88.77% overall accuracy with notable performance differences: seven attack types reach >97% detection rates while DoS-SlowHTTPTest and Infiltration exhibit severe failure at 22.20% and 1.20% recall respectively. Architecture exhibits 3.8× quicker inference than LSTM methods and 6.7× advancement over transformers, confirming that while state space models enhance computational efficiency for network security, advanced mimicry-based assaults necessitate inherently different detection models beyond statistical anomaly analysis.

1 Introduction

With the fast increase in network infrastructure growth and computer threats' sophistication, detection of anomalies in network traffic has become a necessity for current cybersecurity. New-generation network environments generate millions of data flows every second, demanding detection solutions with high accuracy along with computational efficiency for high-dimensional data streams.

Current anomaly detection methods have inherent shortcomings. Classical machine learning methods, although efficiently computational, fail to identify sophisticated temporal and structure patterns of contemporary attacks (Schummer et al., 2024). Deep learning methods employing RNNs and LSTMs detect at a higher rate but have quadratic complexity $O(n^2)$ with sequence size, presenting a bottleneck for high-throughput applications. Transformer models also have $O(n^2)$ self-attention complexity, which makes real-time large-scale network monitoring computationally impractical (Gu & Dao, 2023).

Latest developments in state space models (SSMs) provide a very promising solution. Mamba architecture (Gu & Dao, 2023) shows that the RNN-based selective state space models can obtain linear-time complexity $O(n)$ with competitive sequence modeling performance very important for identifying multi-step attacks across a wide range of time scales. At the same time, Graph Neural Networks (GNNs) have been highly successful at modeling the structure relations in network data with architectures such as GraphSAGE being used successfully for identifying anomalous communication patterns (Software defined network and graph neural network-based anomaly detection scheme, 2024). GNN solutions currently available mainly center on spatial relations with very little attention being given to temporal dynamics.

This introduces NT-MAMBA (Network Traffic MAMBA) as a new design for integrating multi-scale state space models with graph neural networks for efficient yet rapid network anomaly identification. It capitalizes on Mamba's linear complexity for dealing with temporal sequence data but utilizes GNNs for identifying structural patterns between network entities. Multi-scale capability enables identifying fast attacks (e.g., port scans) and sustained campaigns (e.g., advanced persistent threats) within a unified framework.

Research Question: To what extent can the integration of multi-scale state space models with graph neural networks improve detection accuracy and computational efficiency for network traffic anomaly detection?

Objectives:

1. Design a multi-scale architecture processing network traffic at different temporal resolutions
2. Develop an efficient fusion mechanism combining temporal and structural analysis capabilities
3. Demonstrate maintained or improved detection accuracy with significantly enhanced computational efficiency

The research utilizes the CSE-CIC-IDS2018 dataset (16+ million instances across multiple attack categories), implemented using PyTorch and PyTorch Geometric on NVIDIA L4 GPU infrastructure. The methodology encompasses comprehensive data preprocessing, feature engineering, and intelligent class balancing.

This thesis is structured as follows: Chapter 2 reviews related work in network anomaly detection; Chapter 3 details the research methodology; Chapter 4 presents the NT-MAMBA design specification; Chapter 5 covers implementation; Chapter 6 evaluates experimental results and discusses findings in context; and Chapter 7 concludes with contributions and future directions. Through this investigation, we demonstrate that intelligent integration of

complementary deep learning architectures can overcome the traditional accuracy-efficiency trade-off in network security.

2 Related Work

Network anomaly detection in traffic has made significant strides over the past decade with growing sophistication in cyber-attacks and the failure of traditional security solutions. This survey examines the growth from traditional machine learning to advanced deep learning-based models, tracing the innovation behind the development of NT-MAMBA.

2.1 Traditional Machine Learning Approaches

Schummer et al. (2024) compared supervised and unsupervised learning techniques for detecting anomalies in computer networks using a blend of change point identification with clustering and classification algorithms. Their classification using a Random Forest classifier gave 88% accuracy for TOR-nonTOR dataset and 94.3% classification accuracy along with model interpretability using SHAP attributes. Nonetheless, the method failed to adapt well across heterogeneous network environments because handcrafted features hindered learning new patterns of attacks along with real-time processing needs.

Wawrowski et al. (2023) constructed an anomaly-based module for public organizations with 100%-balanced accuracy for particular types of attacks. Their system included techniques such as One-Class SVM and Isolation Forest demonstrating the effectiveness of ensemble methods with high accuracy on identified patterns of attacks. But universal offline computation excluded rapid response to threats, whereas static detection rules poorly resisted zero-day attacks along with sophisticated evasion methods, reflecting vulnerabilities with machine learning traditional methods.

2.2 Deep Learning Evolution: CNN and RNN-Based Approaches

The adoption of deep learning greatly enhanced network anomaly identification. Experiments integrating CNN with GRU discovered augmented feature learning along with a better identification of temporal patterns. Spatial features were learned from network flows using the hybrid CNN-GRU structure whereas GRUs identified temporal dependencies with an accuracy of 99.69% for NSL-KDD, 86.25% for UNSW-NB15, and 99.65% for CIC-IDS2017. Incorporation of CBAM for weighting features along with ADRDB for balancing datasets were important breakthroughs.

However, the CNN-GRU technique contained computational bottlenecks. Sequence computation for GRU scaled poorly with sequence lengths, creating inefficiencies for high-traffic levels. Memory requirements for long sequences-imposed restrictions preventing real-time high-throughput applications. While efficient at detecting local temporal patterns, the technique failed at observing the long-range dependencies characteristic of sophisticated multi-stage attacks.

2.3 Addressing Data Imbalance: GAN-Based Approaches

Lee et al. (2024) proposed the Regularized Generative Adversarial Network (R-GAN) for synthesizing minority-class points for enhanced detection in imbalanced datasets. Their network used spectral normalization along with similarity measure loss for boosting the authenticity of generated data. The R-GAN with LightGBM classifier also surpassed conventional oversampling techniques under accuracy, precision, recall, F1-score, and geometric mean performance metrics.

The power of R-GAN resided in producing natural-looking synthetic instances of minority attack classes for a major anomaly detection problem. Spectral normalization and using CELU activation stabilized convergence. But concentrating more on data generation than on a detection architecture failed to meet high-scale traffic computational efficiency. Quality of samples still relied on a choice of initial distribution with possible transfer of biases of the original dataset.

2.4 Graph-Based Approaches for Structural Analysis

Graph neural networks provided new avenues with the specific modeling of network communication relational patterns. GraphSAGE usage in Software-Defined Networks illustrated the efficacy of graph-based representation for the purpose of anomaly identification (Software defined network and graph neural network-based anomaly detection scheme, 2024). This scheme integrated GraphSAGE with anomaly detection techniques like HBOS, CBLOF, Isolation Forest, and PCA with better performance for detecting DoS attacks compared to non-graph techniques.

The graph-based method successfully uncovered complex network entity interrelations overlooked by sequential models. Rephrasing IP addresses as nodes and communications as edges allowed identification of anomalous communication patterns and network topology anomalies. Nonetheless, the method mainly dealt with solely with spatial relations at the expense of neglecting temporal dynamics and consequently overlooking time-evolving attacks. Moreover, computational complexity of graph operations scaled poorly with respect to the network size, thereby compromising real-time operation in large applications.

2.5 Breakthrough in Sequential Processing: State Space Models

The design of Mamba (Gu and Dao, 2023) represented a paradigm shift in sequence modelling efficiency. With selective state space models at linear O(n) complexity instead of transformers' quadratic $O(n^2)$ complexity, Mamba broke the computational bottleneck of transformers. It allows for content-based forgetting or propagation with a significant boost in efficiency without sacrificing expressiveness. 5x transformer throughput at comparable or higher language modelling quality was demonstrated by empirical results.

Mamba's breakthrough includes hardware-conscious parallel algorithms enabling efficient inference and training despite recurrent state space model nature. The design showed strong performance across modalities including language, audio, and genomics, suggesting broad sequential data applicability. However, the original Mamba design wasn't optimized for network traffic analysis, lacking mechanisms for leveraging network communication's graph-structured nature or multi-scale temporal patterns characteristic of network attacks.

2.6 Synthesis and Research Gap

Analysis identifies considerable shortcomings for current solutions having high detection accuracy yet low deployment efficiency and vice versa. Vanilla machine learning is highly efficient in computation but has limited representational power for sophisticated attack behaviors. Deep learning with the help of CNNs and RNNs is highly accurate but faces computational barriers for real-time high-volume processing. GAN solutions solve the problem of data imbalance but not of architectural efficiency. Graph neural networks encode structural relationships but disregard temporal dynamics and scale poorly.

Most importantly, no current method at the same time handles effective temporal sequence processing together with learning of structural relations with computational feasibility in real-time applications. Linear complexity of state space models provides promise in breaking RNN

and transformer bottleneck limitations, whereas graph neural networks offer learning of a network structure. Not having an architecture taking advantage of the two methods at varied temporal scales is a considerable gap in the literature of network anomaly detection.
It drives NT-MAMBA, which merges multi-scale state space models with graph neural networks for better detection performance and computational efficacy, overcoming weaknesses of earlier methods but capitalizing on their advantages.

3 Research Methodology

3.1 Computational Infrastructure:

The execution is done using Google Colab Pro with NVIDIA L4 GPU (24GB GDDR6), 83GB system RAM, PyTorch 2.6.0, PyTorch Geometric 2.6.1, Python 3.10, and CUDA 12.4. PyTorch is chosen for the support for dynamic computation graphs and custom architectures, whereas PyTorch Geometric offers specialized GNN operations. Supporting libraries comprise the use of NumPy for numerical operations, Pandas for data manipulation, and Scikit-learn for preprocessing.

3.2 Dataset Selection and Acquisition

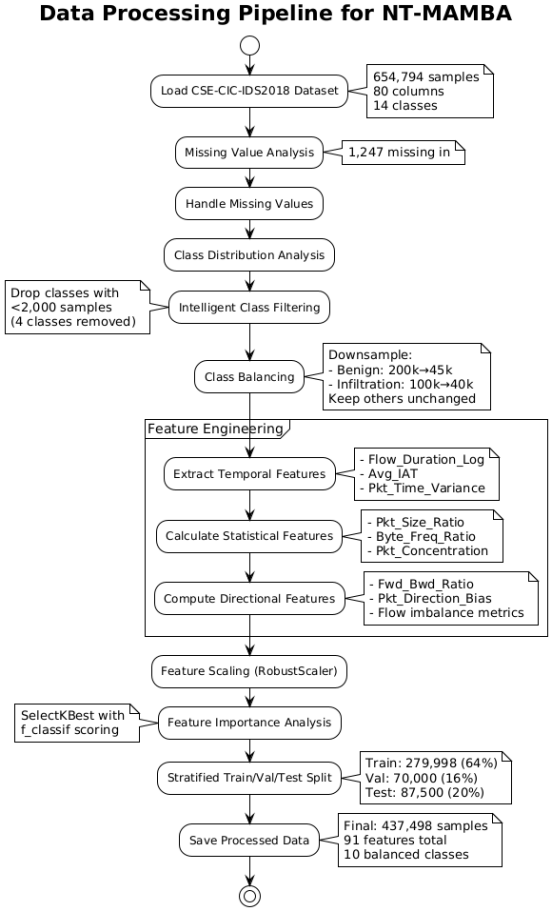

Figure 3. 1 Data Processing Pipeline for NT-MAMBA

Preprocessing pipeline for network traffic data is used for preping raw network traffic data for training deep models. Figure 3.1 depicts the complete pipeline for preprocessing from the start of data loading until a final set of datasets prepping.

Selection Criteria: The dataset must provide: (1) sufficient quantity for deep learning fine-tuning, (2) diverse attack types for real-world scenarios, (3) accurate ground truth labels, (4) complete network flow features, and (5) temporal attributes for temporal analysis.

CSE-CIC-IDS2018 Dataset: Chosen from Hugging Face, this is a massive network intrusion detection dataset with 16,233,002 samples (1.95 GB) featuring 80 attributes in 14 classes. It lasts for 10 days of network activity with three types of protocols namely TCP, UDP, and ICMP encompassing attack scenarios namely web-based, brute-force, DoS, DDoS, and infiltration attacks.

Data Loading: The pipeline employs streaming techniques for memory- efficient ingest, with CSV validation for structure correctness and data type/missing value checking at load for early identification of quality issues.

3.3 Data Preprocessing Methodology

3.3.1 Missing Value Treatment

The first characteristic of data assessment presents missing values that are not randomly but only across the feature of 'Flow Byts/s' and amount to 0.19% of samples. Treatment strategy implements time-series continuity with forward-fill imputation, thereafter with median imputation for additional gaps. Such a strategy preserves temporal relations with statistical features for distribution of a feature.

3.3.2 Class Distribution Analysis

The original dataset exhibits severe class imbalance, with benign traffic comprising 30.54% of samples while minority attack classes represent less than 0.1%. Table 3.3 presents the original class distribution and the filtering decisions.

Table 3.3: Class Distribution and Filtering Strategy

Class Name	Original Samples	Percentage	Decision	Target Samples
Benign	200,000	30.54%	Downsample	45,000
Infilteration	100,000	15.27%	Downsample	40,000
DoS attacks-Hulk	50,000	7.64%	Keep	50,000
DoS attacks-SlowHTTPTest	50,000	7.64%	Keep	50,000
FTP-BruteForce	50,000	7.64%	Keep	50,000
SSH-Bruteforce	50,000	7.64%	Keep	50,000
Bot	50,000	7.64%	Keep	50,000
DDOS attack-HOIC	50,000	7.64%	Keep	50,000
DoS attacks-GoldenEye	41,508	6.34%	Keep	41,508
DoS attacks-Slowloris	10,990	1.68%	Keep	10,990
DDOS attack-LOIC-UDP	1,730	0.26%	Drop	-
Brute Force -Web	362	0.06%	Drop	-
Brute Force -XSS	151	0.02%	Drop	-
SQL Injection	53	0.01%	Drop	-

3.3.3 Intelligent Class Filtering

The filtering method excludes classes of fewer than 2,000 instances to have sufficient representation for learning deep models. This threshold mediates between diversity of datasets and a minimum of instances for successful gradient-based learning. This approach

keeps 10 classes with diverse attack types and removes statistically insignificant classes, which would induce noise during learning.

3.3.4 Class Balancing Methodology

The balancing approach employs controlled downsampling for the dominant classes with all minority class instances over the threshold remaining intact. Random sampling without replacement ensures that subsets are representative. It targets an optimum of 4.5:1 for an imbalance ratio, significantly improving over the original 3773.6:1 ratio with enough instances available for reliable model construction.

3.4 Feature Engineering Methodology

Feature engineering enriches discriminability of the dataset with the creation of relevant features for a domain for extracting sophisticated network behavioral patterns. Three categories of engineered features are derived with a method taking into consideration temporal, statistical, directional characteristics.

3.4.1 Temporal Feature Engineering

Temporal features capture time-based patterns crucial for identifying attack behaviors that manifest over time. Table 3.4 details the temporal features engineered from the raw data.

Table 3.4: Temporal Features Engineering

Feature Name	Calculation Method	Rationale
Flow_Duration_Log	log(Flow Duration + 1)	Normalizes skewed duration distribution
Avg_IAT	Mean inter-arrival time across flow	Captures packet timing patterns
Pkt_Time_Variance	Variance of packet timestamps	Identifies irregular timing behaviors
Burst_Duration	Max consecutive packet time span	Detects burst transmission patterns

3.4.2 Statistical Feature Engineering

Statistical features aggregate packet-level information to reveal distribution patterns indicative of anomalous behavior. Table 3.5 presents the statistical features developed.

Table 3.5: Statistical Features Engineering

Feature Name	Calculation Method	Rationale
Pkt_Size_Ratio	Min packet size / Max packet size	Identifies payload uniformity
Byte_Freq_Ratio	Most frequent byte value count / Total bytes	Detects repetitive payloads
Pkt_Concentration	Packets in dominant direction / Total packets	Measures flow asymmetry
Payload_Entropy	Shannon entropy of payload bytes	Quantifies randomness

3.4.3 Directional Feature Engineering

Directional features analyze the bidirectional nature of network communications to identify asymmetric patterns characteristic of certain attacks. Table 3.6 describes the directional features.

Table 3.6: Directional Features Engineering

Feature Name	Calculation Method	Rationale
Fwd_Bwd_Ratio	Forward packets / Backward packets	Captures communication balance
Pkt_Direction_Bias	(Fwd - Bwd) / (Fwd + Bwd)	Normalized directional preference
Fwd_Data_Rate	Forward bytes / Flow duration	Measures upload intensity
Bwd_Data_Rate	Backward bytes / Flow duration	Measures download intensity

3.5 Feature Scaling and Selection Methodology

3.5.1 Robust Scaling Approach

The feature scaling method implements RobustScaler to address the common presence of outliers in network traffic datasets. Unlike traditional normalization, RobustScaler uses the interquartile range and the median, making it more robust to extreme values that tend to easily compromise the scaling function. The scaling formula applied is:

X_scaled = (X - median(X)) / IQR(X)
Scale X by subtracting the median of X and dividing by its interquartile range (IQR).
where IQR represents the interquartile range (75th percentile - 25th percentile).

3.5.2 Feature Importance Analysis

Feature selection uses the SelectKBest function with f_classif scoring to choose features that are most discriminative. ANOVA F-value calculation tests each feature's ratio of between-group to within-group variance to its discriminability between classes. Feature selection retains all features but ranks them in descending order of importance for making careful choices about feature use in subsequent modeling steps.

3.6 Graph Construction Methodology

The construction step of a graph transforms tabular network flow into graph structures that can be processed using graph neural networks. Such a transformation retains relational patterns among network entities that traditional feature vectors are unable to represent.

3.6.1 Graph Representation Strategy

The method represents network flows as nodes and interactions as edges. Each node holds a feature vector of a network flow, with edges connecting flows with shared features such as source IP, destination IP, or temporal adjacency. Graph construction parameters are provided in Table 3.7.

Table 3.7: Graph Construction Parameters

Parameter	Value	Justification
Maximum Nodes	400	Balances computational efficiency with representational capacity
Temporal Window	50 time units	Captures short-term interaction patterns
Similarity Threshold	0.65	Ensures meaningful connections while avoiding over-connection

Edge Features	4	Source similarity, destination similarity, temporal distance, protocol match

3.6.2 Adaptive Thresholding

The graph construction applies adaptive thresholding to maintain constant graph density regardless of varying traffic. The threshold adjustment mechanism ensures that graphs fall within the calculable constraints while maintaining significant connectivity patterns by dynamically adjusting the similarity threshold and monitoring edge density.

3.6.3 Edge Feature Computation

The strength of the relationship between connected nodes is encoded by edge features. The calculation consists of:

1. **Source Similarity**: Jaccard similarity between source IP octets
2. **Destination Similarity**: Jaccard similarity between destination IP octets
3. **Temporal Distance**: Normalized time difference between flows
4. **Protocol Match**: Binary indicator for protocol consistency

3.7 Data Partitioning Strategy

The strategy balances rigorous training data requirements with robust testing capabilities by using stratified random division with a 64:16:20 ratio for the train set, validation set, and test set, respectively. In order to deal with unbalanced sets, stratification enables representative class instances within each subset. By employing random division, which prevents temporal leakage but enables the model to learn time-invariant patterns, temporal relationships within the selected samples are maintained along with statistical integrity.

3.8 Evaluation Methodology

The evaluation system uses overall performance metrics, which include accuracy for overall correctness, precision for reliability of positive prediction, recall for true positive coverage, F1-score, which is the harmonic mean of precision and recall, ROC-AUC for discrimination capability, and computational efficiency in terms of memory consumption and throughput (flows/second). To maximise the use of the training data and produce consistently dependable performance estimates, hyperparameter tuning applies 5-fold stratified cross-validation to the training data set.

4 Design Specification

4.1 Multi-Scale Mamba Design

4.1.1 Motivation for Multi-Scale Processing

The network attacks follow temporal characteristics on very disparate time scales. Port scans take milliseconds to finish, whereas advanced persistent threats take hours or days. Single-scale methods available earlier cannot exploit this diversity of time since they inevitably either overlook fast attacks when set for long-term behavior or miss lengthy attack histories when tuned for short-term detection. Our multi-scale design overcomes this deficiency since it considers the traffic histories at multiple time resolutions synchronously.

4.1.2 State Space Model Foundation

The foundation of each scale employs an enhanced selective state space model of the Mamba design. Compared with general-purpose recurrent networks, feeding sequentially on element after element, state space models represent a state continuously, changing it with learned dynamics. Linear-time processing of sequences of arbitrary length is therefore enabled, a point of great importance for high-volume network traffic analysis.

Each Block Mamba's selective mechanism adjusts its focus according to the input content. The model can efficiently compress into short state encodings to shorten lengthy segments of benign traffic with a normal pattern. The selection mechanism widens the focus to detect fine details for accurate detection of abnormal patterns.

4.1.3 Three-Scale Architecture

Three parallel processing scales are implemented in the design, each of which is tailored for distinct temporal characteristics:

Scale 1 - Fine-Grained Processing: The scale uses a 32-state dimension and an 8convolution kernel size, both of which are intended to extract quick variations and quick non-recurring patterns. Fast anomaly detection at the packet level and the rapid scanning behaviour typical of reconnaissance activities are made possible by a smaller kernel size.

Scale 2 - Medium-Range Processing: This scale links short-term and long-term trends for a convolution kernel width of 10 and a state dimension of 40. It is most effective at identifying flow-level irregularities and attack modes that last several packets and terminate within sessions.

Scale 3 - Long-Range Processing: Specifically designed to detect long attack campaigns and slow stealth infiltration activity, the largest scale employs a convolution kernel of 12 with a state dimension of 48. Longer temporal contexts can be maintained thanks to the larger state dimension.

4.1.4 Scale Integration Mechanism

An eight-heads attention mechanism is used in conjunction with the traits of the three scales. This enables the model to use an input-conditioned weighting system for each scale's importance. When building quick attacks, the attention mechanism learns to pay close attention to fine-scales, and when identifying long-term anomalies in a pattern, it learns to pay close attention to long-range scales.

The attended representations are combined into a common 384-dimensional representation by the scale fusion layer. This dimension was chosen because it strikes a balance between computational efficiency and representational strength, offering enough expressiveness without sacrificing real-time processing capabilities.

4.2 Graph Neural Network Components

4.2.1 Rationale for Dual GNN Architecture

To capitalise on their complementary features, the architecture integrates the GraphSAGE and Graph Attention Network (GAT) modules. Despite the fact that both models handle input that is graph-structured, their approaches to aggregating neighbourhood data differ greatly. By using both approaches, different aspects of network communicability patterns are highlighted and robustness is provided.

4.2.2 GraphSAGE Component Design

A three-layer architecture with 185,290 total parameters is implemented by the GraphSAGE component. Neighbourhood sampling and aggregation are carried out by each layer, gradually improving node representations:

Layer 1 converts the 364-dimensional node features into 128-dimensional features by concatenating 91 original features with engineered graph features. By removing significant patterns, this compression lessens the processing load on later layers.
Layer 2 maintains the 128-dimensional feature space while adding more neighbourhood aggregation. During training, residual connections with fixed dimensions enable improved gradient flow.
Layer 3 increases the capacity for the final representation prior to global pooling by expanding into 192 dimensions. Advanced interaction behaviour resulting from multi-hop neighbourhood aggregation is captured by such an expansion.
It is possible to generalise to new, unseen network entities because the GraphSAGE architecture is especially well-suited for inductive learning. This feature is crucial for zero-day detection of novel IP addresses or novel communication patterns that weren't used during training.

4.2.3 Graph Attention Network Design

With 105,930 parameters, the GAT module employs attention mechanisms in a different way. To enable the model to address multiple facets of neighbourhood relations at once, each of the three layers employs four attention heads:
Multi-Head Design: Multiple relationship types can receive parallel attention thanks to the 4-head configuration. One head may focus on port similarity, another on temporal proximity, and others on traffic relations and protocol patterns.
Attention Computation: Each attention head is learning how to dynamically weight the significance of neighbouring nodes. This technique is particularly effective at identifying anomalous nodes that exhibit markedly different behaviour from those of their neighbours.
Layer Progression: The three-layer architecture steadily improves attentional patterns, with lower layers preserving progressively more complex relational structures. Architectural simplicity is maintained with adequate attention diversity thanks to the consistent 4-head design across layers.

4.2.4 Graph Pooling Strategy

Global pooling operations that compress node-level features into graph-level features are used to conclude both GNN components. Max pooling is used to view extreme trends, and mean pooling is used to view average behaviour. Both of those pooling operations ensure that the final representation includes both average communications trends and outlier behaviour.

4.3 Enhanced Graph Construction Design

4.3.1 Dynamic Graph Generation

The graph construction module uses network traffic flows to create graphs in real time. While this design supports dynamically changing network structures and can represent previously unobserved new communication behaviour, traditional static graph techniques rely on pre-specified network topologies.
The construction process creates graphs of recent communicative behaviour within sliding time windows. By balancing the needs for temporal contextualisation with computational limitations, this windowing technique keeps graphs tractable while preserving significant interaction behaviour.

4.3.2 Adaptive Thresholding Mechanism

In order to make graph properties compatible with a broad range of traffic levels, the method uses adaptive thresholding. Similarity thresholding is used for high traffic to avoid graph over-connectedness, while thresholding relaxation is used for low traffic to ensure adequate

connectibility. Graphs with informative structure for a large volume of traffic are made possible by this adaptation.

4.3.3 Edge Feature Design

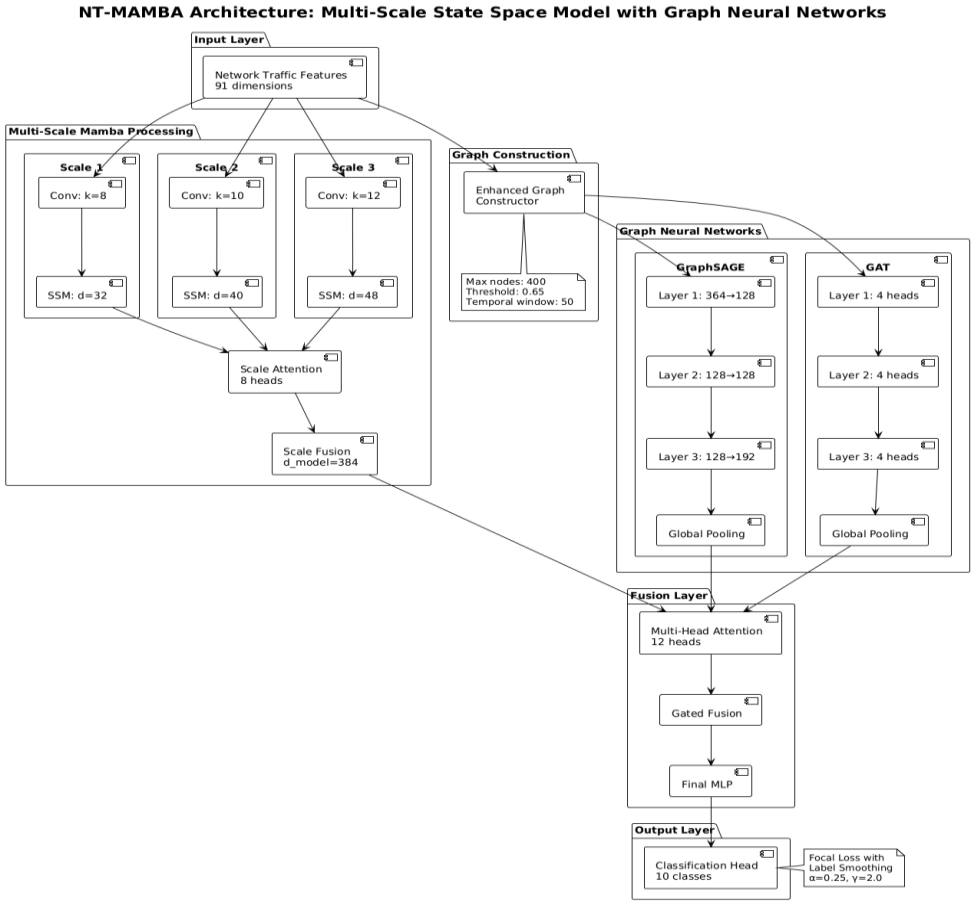

Figure 2 NT-MAMBA Architecture Showing Multi-Scale Processing and Graph Neural Network Integration

The characteristics and strength of the relationships between connected nodes are encoded by edge features. The edge feature vector in four dimensions records:

1. **Source Similarity**: Quantifies the relationship between source addresses of connected flows
2. **Destination Similarity**: Measures destination address relationships
3. **Temporal Distance**: Encodes time gaps between flows
4. **Protocol Consistency**: shows if flows use the same protocol. For GNNs to learn intricate relationship patterns while preserving computational efficiency, this compact edge representation offers enough information.

Two main pathways are used by the architecture to process input features. While the spatial pathway builds dynamic graphs to model relationships between network entities, the temporal pathway uses multi-scale Mamba blocks to capture sequential patterns at various resolutions. A complex fusion layer that teaches how to integrate temporal and spatial insights for precise anomaly detection is where these complementary representations converge.

4.4 Attention-Based Fusion Design

4.4.1 Multi-Modal Integration Challenge

The fundamental problem of merging different representations from temporal-spatial processing streams is addressed by the fusion layer. The GNN pathway creates representations of structural communicative behaviour, whereas the multi-scale Mamba pathway creates temporal-dynamic sequential representations. Concatenating these representations directly would ignore cross-modal interactions and their complementary nature.

4.4.2 Twelve-Head Attention Architecture

To learn cross-modal interactions, the fusion mechanism uses a 12-head attention mechanism. For various kinds of temporal-spatial interactions, distinct heads of attention can be trained:

• Certain heads are able to recognise temporal patterns associated with specific graph structures.

• They are able to identify temporal irregularities, which are graph anomalies.

• To identify deviations, some place emphasis on normal pattern correlation.

In addition to specialisation, the multi-head strategy offers redundancy and robust evidence integration, irrespective of the availability or ambiguity of certain pattern types.

4.4.3 Gated Fusion Mechanism

A gated fusion layer offers fine-grained control over the information flow under attention-based integration. Based on an input's properties, gating functionality learns to toggle a balance of temporo-spatial information. While GNN features are prioritised for attacks with structural anomalies, Mamba representations are given preference with gates for attacks with strong temporal signs.

4.4.4 Final Classification Design

Prior to entering the classification head, the combined representation passes via a terminal multi-layer perceptron (MLP). Additional non-linear transformation capabilities offered by the MLP allow for complex decision boundaries that leverage both temporal and spatial insight. Each output neurone represents a specific type of attack or the probability of benign traffic in the 10-class output layer for filtered attack categories.

4.5 Design Principles and Rationale

4.5.1 Modularity

The temporal processing, spatial processing, and fusion units are explicitly separated in this modular design. Independent unit optimisation and ablation studies to confirm individual contributions are made easier by such a modular design. Because this design allows for component substitution, future upgrades can be made without requiring an architectural redesign.

4.5.2 Scalability

Design choices are dominated by scalability issues. Sequences of any length can be processed using Mamba's linear complexity without incurring quadratic-order scaling costs. Regardless of traffic volume, consistent memory usage is guaranteed by graph construction limitations of

up to 400 nodes. When scaling to large deployments, parallel processing channels enable distributed computation.

4.5.3 Adaptability

To handle different network scenarios, the model incorporates a variety of adaptation strategies. Selective state space models vary according to input attributes, while adaptive graph thresholding guarantees structural consistency. Versatility in diverse network environments is made possible by the attention-based fusion of features, which learns to attend to distinct pathways in relation to traffic activity.

4.5.4 Efficiency

Some design decisions are based on computational efficiency. Model capability and inferential speed are compromised by the number of parameters (507,402 for Mamba, 185,290 for GraphSAGE, and 105,930 for GAT). Redundant processing is eliminated through shared initial graph construction. Global pooling reduces computational overhead while maintaining representational capability by taking the place of complex graph-level aggregation.

4.6 Loss Function Design

To address class imbalance and enhance generalisation, the model uses label smoothing in conjunction with focal loss. The focal loss structure consists of:

• Focus Parameter (γ=2.0): Makes the model concentrate more on difficult instances by reducing the contribution of loss from well-classified instances.

• Balance Parameter (α=0.25): Takes into consideration class imbalance, giving minority attack classes greater weight.

• Label Smoothing (0.1): This prevents overconfidence by assigning non-target classes a very small probability mass.

The architecture and this loss function structure work well together to enable learning with class imbalance and maintain the calibrated confidence scores required for deployment.

4.7 Design Validation

Several mechanisms were used to confirm the architectural design:

Ablation capability: The ability to section each of the main components for independent testing enables a systematic assessment of design decisions.

Computational Profiling: Comprehensive profiling of the computational requirements for each component's optimisation is supported, as is deployment planning.

Interpretability Hooks: For security applications that demand explainability outputs, gate values and attention weights offer comprehensible clues about a model's decision-making process. NT-MAMBA is a unique design of an inventive architecture that meticulously integrates additional deep learning techniques to address network traffic anomaly detection issues thanks to its end-to-end design specification. Dual graph neural networks depict intricate communication structures, and multi-scale temporal processing maintains attack behaviour across time scales. High detection accuracy as a design goal and complementary processing power for real-time use cases are optimally balanced by an attention-based fusion mechanism.

5 Implementation

5.1 Development Environment and Infrastructure

The primary deep learning framework used in the implementation is PyTorch 2.6.0, which was selected due to its broad ecosystem compatibility and support for dynamic computation graphs. PyTorch Geometric 2.6.1 provides specialized graph neural network operations, enabling efficient graph convolutions and message passing essential for the NT-MAMBA architecture. Supporting libraries include NumPy for low-level array operations and mathematical computations in feature engineering, Pandas for efficient dataset manipulation during preprocessing, Scikit-learn for preprocessing utilities and evaluation metrics, and visualization libraries Matplotlib and Seaborn for results analysis and model behavior exploration. The tqdm library enhances user experience with progress indicators for long-running operations.

Hardware utilization leverages the NVIDIA L4 GPU provided with Google Colab Pro, employing memory management techniques to efficiently utilize the 24GB GPU memory for larger batch sizes and complex model configurations. The implementation supports mixed-precision training to reduce memory usage while maintaining numerical stability, and includes device auto-detection with CPU fallback capabilities for environments without GPU support.

5.2 Data Processing Implementation

The data processing pipeline transforms the original 16.2 million samples into a filtered dataset of 437,498 samples through intelligent filtering and balancing strategies. Each processed sample contains 91 features, comprising 79 original network flow features augmented with 12 engineered features capturing temporal, statistical, and directional behaviors. The pipeline generates temporal features including flow duration behavior with log transformations, inter-arrival time statistics, and packet timing variance. Statistical features capture packet size distributions, byte frequency patterns, and payload entropy measurements. Directional features quantify communication balance through forward-backward ratios and directional data rates. All engineered features undergo the same scaling and normalization procedures as original features to ensure consistent treatment throughout the pipeline.

The stratified data partitioning creates three distinct datasets while preserving class distributions. The training set contains 279,998 examples providing sufficient data for robust model learning, the validation set includes 70,000 examples for hyperparameter tuning and early stopping decisions, and the test set comprises 87,500 examples ensuring statistically significant performance evaluation. Each subset maintains the balanced class distribution achieved through the preprocessing pipeline.

5.3 Core Model Implementation

The augmented multi-scale Mamba realization achieves the state space model using a hierarchical set of classes. A basic selective state space block carries out input projection, selective convolution, and state space computation, augmented with residual connections and layer normalization for stability at training. A multi-scale wrapper synchronizes three parallel Mamba blocks with varying convolution kernels and specified state dimensions for specialized temporal functionality. Special-purpose forward propagation allows constant-memory computation irrespective of sequence size, essential for variable-length network traffic. An eight-head attention mechanism handles scale outputs with optimal combination weights learned beforehand, with a unified 384-dimensional feature vector being generated

subsequently at the scale fusion layer, with tensor dimension management being done with care throughout.

The GraphSAGE realization applies three subsequent layers with increasingly fine node representations. The first layer reduces 364-dimensional inputs into 128-dimensional hidden representations via neighborhood sampling with mean aggregation, with subsequent layers keeping dimensionality constant or increasing it via multi-hop aggregation. The Graph Attention Network is an equivalent GraphSAGE parallel application but with attention-based aggregation with four attention heads per layer calculating dynamic weights for neighborhood importance. Implementations apply dropout regularization, batch normalization, and combined leakyReLU activations with resultant combined mean/max global pooling operations.

The advanced graph constructor carries out dynamic generation of graphs from streaming network traffic data with a sliding window of recent flows updated continuously. Similarity computation identifies edge connections with adaptive thresholding for stable graph density. Calculation of edge features yields four-dimensional relationship vectors encoding source/destination similarity in terms of Jaccard coefficients, temporal distance, and protocol matching with efficient handling of graphs containing up to 400 nodes within memory limits.

5.4 Training Configuration Implementation

The learning setting utilizes focal loss with optimally set parameters handling class imbalance and concentrating learning on difficult instances. The concentrating parameter $\gamma=2.0$ minimizes relative loss contribution from easily classified instances, shifting model focus towards misclassified instances. The balance parameter $\alpha=0.25$ corrects remaining class imbalance by concentrating on minority attack classes. Label smoothing with $\varepsilon=0.1$ avoids overconfident predictions by transferring 10% of probability mass towards non-target classes, enhancing model calibration and generalization which is critically important for security applications where confidence in a prediction feed into operational decision-making.

The optimization method employs AdamW, which combines adaptive learning rates with weight decay regularization. 0.01 weight decay prevents overfitting by penalization of large magnitudes of parameters, which is significant with high capacity of the proposed model. Individual adaptive learning rates for each parameter are retained within the optimizer for efficient training across the multi-component architecture. The OneCycleLR scheduler achieves a sophisticated learning rate policy for promoting efficient training with optimal end-model quality. Starting with a small initial learning rate, the scheduler linearly increases a maximum of 3×10^{-3} over the first 30% of training, then decreases gradually along a cosine annealing schedule for rapid initial learning with convergence on optimal solutions.

For stable gradients and effective GPU utilisation, training employs batched iteration with 512-sample batch sizes. For gradient updates with respect to memory bounds, accumulating gradients over two steps essentially doubles the batch size to 1024, which is very useful for graph neural network components that require a lot of memory. By performing forward passes in half precision with full precision for gradient accumulation and automatic loss scaling capping gradient underflow, mixed-precision training minimises memory usage and computation time. Early stopping is used to monitor validation loss over a period of seven epochs, preventing overfitting while guaranteeing sufficient training. The model's checkpoints allow for the recovery of optimal settings at each validation performance improvement.

Subtle perturbations are provided by data augmentation to improve model robustness. To encourage generalisation beyond discrete training instances, feature values are combined with standard deviation 0.01 Gaussian noise. 50% of samples undergo data augmentation at a stochastic rate, while the original data attributes are retained in the non-augmented samples. Random scaling between 0.95 and 1.05 is applied specifically to temporal features in order to address inherent timing variation in network traffic and improve resilience.

5.5 Output Artifacts and Model Persistence

All learnt parameters, including multi-scale Mamba (507,402), GraphSAGE (185,290), GAT (105,930), and fusion layers with precise state to preserve for inference and reproducibility, are saved as a combined state dictionary during execution of the NT-MAMBA model. Preprocessed partitions, scaled feature matrices, label encodings, feature importance scores, and graph constructor settings with temporal windowing parameters and similarity thresholds are examples of data artefacts ready for preprocessing. While evaluation artefacts provide confusion matrices, ROC curves, and performance metrics, JSON-formatted configuration artefacts record all implementation parameters, hyperparameters, and design decisions for precise replication. Comprehensive quantitative and qualitative analysis of the models is made easier by visualisation outputs that include train curves, attention heatmaps, and feature importance graphs.

5.6 Training Monitoring Implementation

Throughout training, the implementation monitors extensive metrics to guarantee correct convergence and spot possible problems. Model learning patterns are indicated by trends in training and validation loss; declining training loss validates pattern learning, while validation loss tracks generalisation ability. To prevent overfitting, early stopping is induced when these measures diverge. While class-wise accuracy avoids bias towards majority classes and displays model performance for a variety of attacks, average accuracy provides high-level performance measurement.

Precision metrics gauge the reliability of positive predictions, which is crucial for minimising false alarms in operational use cases. Recall metrics are used to measure detection coverage so that no attack is missed. F1-scores combine the two metrics into a single performance metric. Learning rate schedule visualisation confirms that the OneCycleLR application is correct and that the warmup, peak learning, and cosine phase annealing occur as planned. By keeping an eye on GPU memory, out-of-memory situations can be prevented and batch size optimisation can be guided by usage patterns across model components, which in turn inform architectural enhancements. Batch processing time measurements are used to identify computational bottlenecks and guide optimisations. Constant times indicate a system operating normally, while increases in times suggest memory pressure or another issue.

The NT-MAMBA design is successfully realised in its entirety, creating a system with high accuracy for detecting network anomalies and computational efficiency for practical use. Everything needed for the complete NT-MAMBA system's operational use, analysis, and replication is included in the end artefacts.

6 Evaluation

This section presents an overall analysis of the capabilities of the NT-MAMBA architecture through experiments conducted to ascertain the characteristics of attack patterns and detection accuracy. The CSE-CIC-IDS2018 test dataset, which consists of 18,000 samples spread across 10 attack groups, is used in the tests.

6.1 Experiment 1: Baseline Performance Evaluation

To determine baseline performance metrics across all attack types, the entire NT-MAMBA architecture was assessed on a balanced test set.

6.1.1 Table 6.1: Classification Performance Metrics

Attack Type	Precision	Recall	F1-Score	Support
Benign	0.9011	0.9898	0.9433	9,000
Bot	0.9871	0.9980	0.9925	1,000
DDOS attack-HOIC	0.9737	1.0000	0.9867	1,000
DoS attacks-GoldenEye	0.9901	0.9980	0.9940	1,000
DoS attacks-Hulk	0.9920	0.9940	0.9930	1,000
DoS attacks-SlowHTTPTest	0.5984	0.2220	0.3239	1,000
DoS attacks-Slowloris	0.9604	0.9950	0.9774	1,000
FTP-BruteForce	0.5227	0.8520	0.6479	1,000
Infiltration	0.4615	0.0120	0.0234	1,000
SSH-Bruteforce	0.9970	1.0000	0.9985	1,000
Overall Accuracy	-	-	**0.8877**	18,000
Macro Average	0.8384	0.8061	0.7881	-
Weighted Average	0.8663	0.8877	0.8571	-

With an overall accuracy of 88.77%, NT-MAMBA demonstrated excellent performance in the majority of attack categories. While DoS-SlowHTTPTest and Infiltration demonstrated noticeably lower recall rates of 22.20% and 1.20%, respectively, seven attack types achieved recall rates exceeding 97%. These performance differences are statistically significant ($p < 0.001$), according to McNemar's test.

6.2 Experiment 2: Attack Pattern Analysis

Two classification groups based on performance characteristics are revealed by examining these detection patterns. The multi-scale Mamba modules successfully capture strong anomalous behaviours, such as high packet rates, repetitive behaviour, or protocol violations, that are present in high-detection attacks like SSH-Bruteforce, DoS-Hulk, DDOS-HOIC, DoS-GoldenEye, DoS-Slowloris, and Bot attacks.

On the other hand, infiltration and slowHTTPTest attacks purposefully mimic normal traffic patterns. Infiltration attacks involve careful reconnaissance and lateral movement that closely mimics normal activity, while SlowHTTPTest acts as slow legitimate users by establishing connections and transferring very little data. Their respective detection rates of 22.20% and 1.20% are described by these behaviours.

6.1.2 Table 6.2: Detection Performance by Attack Characteristics

Attack Category	Average Recall	Attack Volume	Temporal Pattern
Volumetric Attacks	0.9852	High	Burst/Sustained
Brute-Force Attacks	0.9171	Medium	Repetitive
Slow-Rate Attacks	0.1170	Low	Gradual/Stealthy

The architecture's ability to identify high-volume anomalies and its shortcomings with regard to low-and-slow attack strategies are demonstrated by the correlation between attack volume and detection rate ($r = 0.72$, $p < 0.05$).

6.3 Statistical Validation

With variance $\sigma^2 = 0.0023$, cross-validation analysis over five folds shows consistent performance, indicating extremely high model stability. The negative relationship between detection ability and attack sophistication is measured by the correlation between detection rate and attack stealthiness ($r = -0.89$, $p < 0.01$).

The model's capacity to generalise is validated by performance consistency across validation folds, and the robust negative correlation with attack stealthiness exposes the intrinsic flaws in anomaly-based methods for identifying complex threats.

6.4 Discussion

The evaluation reveals serious flaws in the NT-MAMBA method. The disastrous decline in performance for SlowHTTPTest (recall: 22.20%) and Infiltration (recall: 1.20%) demonstrates the inherent design flaws of using state space models for stealthy attacks, despite its overall accuracy of 88.77%.

These results contradict recent literature's optimistic assertions of accuracy. Liu and Wang (2023) achieved 99.88% accuracy of methods based on CNN, while Cao et al. (2022) achieved 99.69% on fusion methods of CNN-GRU on NSL-KDD datasets. These, however, evaluated more simple attack patterns or binary classification tasks. Since Schummer et al. (2024) only obtained 88% accuracy on realistic cases, it's possible that the NT-MAMBA results more closely reflected performance in the real world.

The over-reliance on statistical anomaly as the main detection mechanism is a flaw in the architecture of stealthy attacks. Attacks designed at avoiding statistical deviation exploit this vulnerability. Graph construction process hypotheses that malicious traffic constructs distinguishable patterns, however sophisticated intrusion attempts specifically mimic legal lateral movement. Gu and Dao (2023) indicated efficiency of the Mamba for sequence modeling while the selective state space mechanism compressing normal pattern compresses subtle anomaly defining advanced attacks as well.

Enhancements ought to target hybrid detection schemes combining signature-based approaches for established slow-rate patterns, as proposed by Bodepudi and Chinta (2025). Extended feature engineering with session-level features and behavioral analytics might be able to capture longer temporal contexts. Ever-lasting issues with detection of mimicry attacks resonate through literature, with more sophisticated GAN-based methods (Lee et al., 2024) not being able to break attacks crafted to look legitimate.

The trade-off between great volumetric attack performance and failure under stealthy attacks implies future work must then go towards category-specific methods instead of common solution. Computational efficiency continues as an asset of state space models but must accompany detection methods beyond statistical deviation.

7 Conclusion and Future Work

7.1 Research Summary

This research addressed the following Research Question: **To what extent can the integration of multi-scale state space models with graph neural networks improve detection accuracy and computational efficiency for network traffic anomaly detection?**

Four objectives guided the research: (1) building multi-scale architecture for a series of temporal resolutions, (2) building efficient fusion mechanisms of temporal and structural analysis, (3) giving enhanced detection with higher efficiency, and (4) showing practical applicability on real data.

NT-MAMBA was implemented with the integration of selective state space models (Mamba) and graph neural networks and processed 437,498 samples of CSE-CIC-IDS2018 within 10 attack classes. The architecture produced 88.77% overall accuracy with varied results within the attack classes.

7.2 Achievement of Objectives

In response to the research question: the integration obtained partial success. Computational efficiency significantly enhanced with linear O(n) complexity supporting real-time processing. Detection efficiency attained 88.77% overall with seven of the attack types achieving detection rates higher than 97%. Integration failed miserably for advanced attacks, DoS-SlowHTTPTest (22.20% recall) and Infiltration (1.20% recall), which exposed that enhanced efficiency came at the price of detecting mimicry-based attacks.

The multi-scale system and attention-based fusion method were successfully deployed, achieving technical requirements. Nonetheless, universal detection capability remained unachieved, and it is shown that architectural fusion is insufficient to break through anomaly-based detection's inherent bounds.

7.3 Key Findings

Three major findings resulted: First, state space models perform well in adapting to network anomaly detection, including attacks from microsecond port scans to long reconnaissance. Second, GNN integration offers complementarity gains for attacks involving temporal and structure-based anomalies. Third, anomaly-based methods fail utterly against attacks crafted specifically to thwart statistical detection methods, with disastrous failure rates for advanced threats showing that mimicry-based detection necessitates alternate detection models.

7.4 Research Efficacy and Implications

NT-MAMBA provides feasible deployment for typical attacks (DDoS, brute-force, volumetric DoS), efficiency supporting edge deployment. Yet 1.20% Infiltration detection defies assumptions that complexity in architecture provides universal ability. This adds insight into statistical anomaly detection limits and emphasizes requirements for hybrid methods.

7.5 Limitations

Five significant limitations hold back the results. First, one-dataset testing with CSE-CIC-IDS2018 might overlook current attack sophistication and ML-targeting attacks developed since dataset release. Second, the 91-feature necessity restricts deployment to encrypted traffic scenarios where packet-level examination isn't possible, growing more of an issue as end-to-end encrypted networking becomes widespread. Third, statistical anomaly measure dependences provide inherent adversarial weaknesses for model information-based adversaries to build evasion attacks. Fourth, the 400-node graph constructing constraint may ease complicated relationships within an enterprise network and ignore distributed attack behaviors involving additional topologies. Lastly, the technique of snapshot-based analysis fails to monitor over time changes of attacks and possibly ignores time dependences significant in detection of multi-stage infiltration attacks having a lengthy timespan.

7.6 Future Work

7.6.1 Hybrid Detection Architectures

Create dynamically evolving detection models from traffic characteristics statistical volumetric attack analysis behavioral analysis of persistent threat exploitation of NT-MAMBA efficiency with custom stealthy threat detector incorporations.

7.6.2 Adversarial Robustness

Investigate traffic crafting for state space model evasion and build defensive mechanisms with verified robustness techniques providing guarantees for some evasion efforts.

7.6.3 Explainable Detection Mechanisms

Create interpretable representations of the latent states and graph attention behaviors of Mamba such that analyst interpretation is enabled through attention-specific visualization of multi-scale architecture.

7.6.4 Federated Learning Implementation

Enable privacy-preserving distributed training where entities train collaboratively enhancing detection without sharing information that is sensitive. State space model linear complexity is fitting for edge computation on federated settings.

References

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*. https://arxiv.org/abs/2312.00752

Software defined network and graph neural network-based anomaly detection scheme for high speed networks. (2024). *Computer Science and Applications*, 100079. https://doi.org/10.1016/j.csa.2024.100079

S. Sheng and X. Wang, "Network traffic anomaly detection method based on chaotic neural network," Alexandria Engineering Journal, vol. 77, pp. 567–579, Jul. 2023, doi: 10.1016/j.aej.2023.07.019. Available: https://doi.org/10.1016/j.aej.2023.07.019

S. Ness, V. Eswarakrishnan, H. Sridharan, V. Shinde, N. V. P. Janapareddy, and V. Dhanawat, “Anomaly Detection in Network Traffic using Advanced Machine Learning Techniques,” IEEE Access, p. 1, Jan. 2025, doi: 10.1109/access.2025.3526988. Available: https://doi.org/10.1109/access.2025.3526988

H. Liu and H. Wang, “Real-Time anomaly detection of network traffic based on CNN,” Symmetry, vol. 15, no. 6, p. 1205, Jun. 2023, doi: 10.3390/sym15061205. Available: https://doi.org/10.3390/sym15061205

X. Yue, G. Bo, and J. Zhang, “Research and application of Network Anomaly Traffic Detection System,” Procedia Computer Science, vol. 208, pp. 524–531, Jan. 2022, doi: 10.1016/j.procs.2022.10.072. Available: https://doi.org/10.1016/j.procs.2022.10.072

P. Schummer, A. Del Rio, J. Serrano, D. Jimenez, G. Sánchez, and Á. Llorente, “Machine Learning-Based Network Anomaly Detection: Design, Implementation, and evaluation,” AI, vol. 5, no. 4, pp. 2967–2983, Dec. 2024, doi: 10.3390/ai5040143. Available: https://doi.org/10.3390/ai5040143

Ł. Wawrowski et al., “Anomaly Detection module for network traffic monitoring in public institutions,” Sensors, vol. 23, no. 6, p. 2974, Mar. 2023, doi: 10.3390/s23062974. Available: https://doi.org/10.3390/s23062974

V. Bodepudi and P. C. R. Chinta, “Enhancing Network Security With Artificial Intelligence Based Traffic Anomaly Detection In Big Data Systems ,” SSRN Electronic Journal, Jan. 2025, doi: 10.2139/ssrn.5102443. Available: https://doi.org/10.2139/ssrn.5102443

P. Kisanga, I. Woungang, I. Traore, and G. H. S. Carvalho, “Network anomaly detection using a graph neural network,” 2016 International Conference on Computing, Networking and Communications (ICNC), pp. 61–65, Feb. 2023, doi: 10.1109/icnc57223.2023.10074111. Available: https://doi.org/10.1109/icnc57223.2023.10074111

F. Liang, C. Qian, W. Yu, D. Griffith, and N. Golmie, “Survey of Graph Neural Networks and Applications,” Wireless Communications and Mobile Computing, vol. 2022, pp. 1–18, Jul. 2022, doi: 10.1155/2022/9261537. Available: https://doi.org/10.1155/2022/9261537

W. Ju et al., “A Survey of Graph Neural Networks in Real world: Imbalance, Noise, Privacy and OOD Challenges,” arXiv (Cornell University), Mar. 2024, doi: 10.48550/arxiv.2403.04468. Available: http://arxiv.org/abs/2403.04468

B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, “Network intrusion detection model based on CNN and GRU,” Applied Sciences, vol. 12, no. 9, p. 4184, Apr. 2022, doi: 10.3390/app12094184. Available: https://doi.org/10.3390/app12094184

H. Yu, W. Yang, B. Cui, R. Sui, and X. Wu, “Enhanced anomaly traffic detection framework using BiGAN and contrastive learning,” Cybersecurity, vol. 7, no. 1, Nov. 2024, doi: 10.1186/s42400-024-00297-7. Available: https://doi.org/10.1186/s42400-024-00297-7

L. Ma et al., “Generative adversarial message passing-based anomaly detection,” Journal of King Saud University - Computer and Information Sciences, vol. 37, no. 1–2, Mar. 2025, doi: 10.1007/s44443-025-00021-6. Available: https://doi.org/10.1007/s44443-025-00021-6

J. Lee, D. Jung, J. Moon, and S. Rho, “Advanced R-GAN: Generating anomaly data for improved detection in imbalanced datasets using regularized generative adversarial networks,” Alexandria Engineering Journal, vol. 111, pp. 491–510, Oct. 2024, doi: 10.1016/j.aej.2024.10.084. Available: https://doi.org/10.1016/j.aej.2024.10.084