CONTENTS

AI-BASED CYBER ATTACK DETECTION IN NETWORK SYSTEMS EMPLOYS DEEP LEARNING TECHNIQUES TO DETECT ANOMALIES AND NETWORK INTRUSIONS IN REAL TIME

Abstract

This project focuses on the design and development of an intelligent, automated Network Intrusion Detection System (NIDS) utilizing advanced Deep Learning (DL) techniques to identify sophisticated cyber attacks, including zero-day exploits, by detecting real-time behavioral anomalies in network traffic. Traditional signature-based systems are fundamentally limited to known threats and struggle with the volume and complexity of modern encrypted traffic. To address these deficiencies, this research proposes leveraging hybrid DL architectures, specifically Bidirectional Long Short-Term Memory (BiDLSTM) networks combined with Convolutional Neural Networks (CNN) or Autoencoders, which are adept at autonomously extracting complex temporal and statistical features from continuous network flow data. The system employs flow characteristics generated from contemporary datasets, such as the UNSW-NB15 and CICIDS2018, using high-performance feature extraction tools. This anomaly-based approach overcomes the inherent limitations of static machine learning models and manual feature engineering. The final architecture is engineered for low-latency, high-throughput environments, requiring dedicated GPU acceleration for inference and high-speed network interface cards (NICs) for packet capture. The system is designed for modularity and seamless integration with existing Security Information and Event Management (SIEM) solutions, transforming raw traffic monitoring into actionable, real-time decision support for comprehensive network defense.

List of Abbreviations

A comprehensive list of acronyms and technical terms relevant to Deep Learning (DL), Network Security, and Intrusion Detection Systems (IDS) is presented below.

List of Key Abbreviations

Acronym	Definition	Domain Reference
AD	Anomaly Detection	Core behavioral detection methodology
AUC	Area Under the Curve	Performance metric for classification models
BiDLSTM	Bidirectional Long Short-Term Memory	Recurrent Neural Network architecture utilizing sequence context
CNN	Convolutional Neural Network	Deep learning network suitable for spatial feature extraction
DDoS	Distributed Denial of Service	Common modern attack type involving flooding a target
DL	Deep Learning	Subset of Machine Learning utilizing Deep Neural Networks (DNN)

| DNN | Deep Neural Network | Multi-layered artificial neural network architecture | | FPs | False Positives | Incorrect flagging of benign traffic as malicious | | HIDS | Host-based Intrusion Detection System | IDS monitoring activity on a single computer or endpoint | | IDS/NIDS | (Network) Intrusion Detection System | Primary device or application for monitoring network activity | | IPS | Intrusion Prevention System | System designed to detect and automatically block attempted attacks | | LSTM | Long Short-Term Memory | Recurrent Neural Network variant for sequential data | | ML | Machine Learning | Broad field of study for algorithmic pattern recognition | | NGFW | Next-Generation Firewall | Advanced firewall using deep packet inspection and intrusion detection | | NIC | Network Interface Card | Critical hardware component for real-time packet capture | | ROC | Receiver Operating Characteristic | Graphical plot illustrating classifier performance | | SIEM

| Security Information and Event Management | Centralized system for integrating NIDS alerts and logs | | SVM | Support Vector Machine | Traditional machine learning classification algorithm | | Zero-Day | Zero-Day Exploit | A novel vulnerability exploited before a vendor patch exists |

Chapter 1: Introduction

The exponential growth of global network connectivity and the integration of diverse devices, particularly within the realm of the Internet of Things (IoT), have drastically expanded the surface area vulnerable to cyber exploitation. Concurrently, malicious actors are leveraging increasingly sophisticated technologies, often incorporating Artificial Intelligence (AI) themselves, to maximize the financial returns from their operations. This dynamic environment places unprecedented pressure on conventional network defense mechanisms, demanding a paradigm shift toward intelligent, real-time, and predictive security solutions. This project addresses the critical need for a next-generation security system capable of detecting subtle behavioral deviations indicative of novel cyber threats.

1.1 Motivation

The necessity for an AI-based NIDS is underscored by the escalating financial and operational impact that cyber threats currently impose on organizations globally. Cybercrime is predicted to cost the world an extraordinary $9.5 trillion USD in 2024, highlighting the scale of the economic damage caused by successful intrusions. The direct financial cost per incident is also increasing significantly; the global average cost of a data breach crossed $4.88 million in 2024, representing a 10% increase compared to the previous year.

Beyond the direct financial losses, current defense mechanisms fail critically in terms of response time. Security teams take an alarming average of 277 days to identify and contain a data breach. This prolonged exposure time provides attackers with extensive opportunity for data exfiltration and persistent network presence. Breaches involving lost or stolen credentials are even more protracted, requiring an average of 328 days to identify and contain. This operational mismatch—where human response speed lags months behind the

rapid, opportunistic nature of financially motivated hackers —proves that automation is no longer optional but a fundamental necessity for survival in the current threat landscape.

Furthermore, the complexity of attacks is soaring. Encrypted threats, which bypass many traditional inspection methods, increased by 92% in 2024. This proliferation of encrypted communication mandates a strategic shift from inspecting payload content (Deep Packet Inspection, or DPI) to monitoring the *behavior* and *patterns* of network flows, requiring advanced learning capabilities that only deep learning can adequately provide. Organizations that extensively use security AI and automation to prevent data breaches realize a substantial average annual cost savings of $2.22 million compared to those that do not. This demonstrated return on investment validates the complex research required for developing this intelligent system, confirming that AI integration is not merely a technical advancement but a critical economic intervention strategy enabling preemptive and sustainable defense.

1.2 Problem Definition

Current network security strategies are fundamentally compromised by the intrinsic technical limitations of legacy detection systems. The first and most significant failure lies with **Signature-Based Intrusion Detection Systems (NIDS)**. These systems function by comparing network packets against a database of known malicious patterns or identifiers. While effective against recognized malware, they are inherently limited to identifying *known* threats, leaving networks completely vulnerable to zero-day attacks—novel exploits that lack existing signatures. Moreover, maintaining and manually updating these signature databases is a resource-intensive task that struggles to scale with the sheer velocity of modern threat evolution.

To counter the zero-day threat, security research introduced **Anomaly-Based Detection (AD)**. However, early AD systems were reliant on statistical modeling or basic heuristics and struggled with significant technical hurdles. Accurately defining what constitutes "normal" behavior is extremely challenging in dynamic, complex network environments. This difficulty often resulted in a high rate of false positives (FPs), where legitimate traffic was incorrectly flagged as malicious. A high volume of FPs leads inevitably to "operator fatigue," where security teams begin to ignore alerts, effectively neutralizing the system's defensive value.

A third limitation stems from conventional machine learning (ML) techniques (such as Support Vector Machines, Random Forests, or k-Nearest Neighbors) when applied to network data. These models require resource-intensive, manual feature engineering. Analysts must spend significant time selecting relevant features and reducing data dimensionality, a process that is slow to adapt and prone to missing complex, non-linear relationships hidden within the high-dimensional network flow data. Supervised ML algorithms, in particular, excel only in identifying *known* attack categories for which labeled training data exists, faltering immediately in the face of true zero-day attacks.

Consequently, there is a critical need for an automated, low-latency system capable of dynamically learning complex behavioral patterns from high-throughput network traffic, achieving high detection accuracy against novel threats while simultaneously minimizing the operational burden of false positives.

1.3 Objective of the Project

The primary objective of this project is to develop a robust, Deep Learning (DL)- based anomaly detection engine capable of classifying continuous network flows into Benign traffic or specific Attack categories (e.g., Distributed Denial of Service (DDoS), Botnet, Exploits, Web attacks) with high confidence and minimal latency.

To achieve this primary goal, the project will fulfill several key technical objectives:

1. **Preprocessing Pipeline Design:** Design and implement a real-time preprocessing pipeline capable of effectively extracting relevant network flow features (such as those generated using tools like CICFlowMeter) and transforming them into sequential, time-series data representations suitable for Recurrent Neural Networks (RNNs).
2. **Model Performance Optimization:** Develop and train a sophisticated Deep Neural Network (DNN) architecture (e.g., BiDLSTM or CNN-LSTM hybrid) optimized to achieve superior classification performance, targeting metrics such as accuracy, precision, and recall, with the goal of exceeding 97% accuracy, particularly ensuring balanced metrics across minority attack classes.

3. **Low-Latency System Engineering:** Engineer a low-latency system architecture utilizing high-speed hardware components (dedicated NICs, GPU acceleration) to ensure the system performs real-time inference at network line speed, a necessity for continuous monitoring and immediate alerting.
4. **Interface and Integration:** Develop a modular output system capable of generating rapid, actionable alerts that can be seamlessly integrated with external Security Information and Event Management (SIEM) systems for comprehensive security operations and forensic documentation.

1.4 Limitations of the Project

While the proposed DL-based NIDS represents a significant advancement in detection capability, its development and deployment are constrained by practical limitations.

First, the system's **computational intensity** presents a significant challenge. Real-time processing of massive network traffic volumes requires substantial investment in high-performance computing resources. Specifically, GPU acceleration is mandatory for efficiently handling the complex calculations involved in deep learning model training and low-latency inference, which may limit accessibility for smaller organizations or those with constrained IT resources.

Second, the model's **generalization capability** is inherently tied to the diversity and quality of the utilized datasets. If the model is trained exclusively on certain network topologies or attack types, it may face challenges generalizing when deployed in a highly unique or novel production environment outside of the training data distribution. Continuous retraining and adaptive learning mechanisms are required, adding complexity to long-term maintenance.

Third, the system is designed strictly as a **Network Intrusion Detection System (NIDS)**, meaning its primary function is to provide timely alerts regarding potential threats. It functions as a decision-support tool, not an automatic mitigation tool, and therefore cannot replace expert security judgment. Full automated defense requires integration with an Intrusion Prevention System (IPS) or a Security Orchestration, Automation, and Response (SOAR) platform.

Finally, although the system excels at detecting anomalous behavior within encrypted flows by analyzing flow characteristics, it remains **unable to inspect the actual payload contents** without an external decryption mechanism (which is often impractical or legally constrained). Therefore, the precise severity and type identification for encrypted attacks relies purely on external flow metrics, potentially limiting the granularity of the threat assessment.

Chapter 2: Literature Survey

2.1 Introduction: Evolution of Intrusion Detection

The history of cybersecurity defense reflects a continuous arms race between attackers and defenders. Intrusion Detection Systems (IDS) serve as the fundamental tool for monitoring network activity to identify unauthorized use, misuse, or policy violations. Historically, IDS methodologies were classified based on their detection paradigm: signature-based or anomaly-based.

Signature-based NIDS, which relies on pattern matching of known malicious signatures, formed the bedrock of early network security. However, as threat objectives became increasingly complex, this reactive approach proved insufficient. The community widely acknowledged the need to move beyond known threats, recognizing that **anomaly-based detection (AD)** offers higher efficiency and dynamic adaptability. AD approaches monitor system activity, building a profile of "normal" behavior, and then classifying any significant deviation from this profile as an anomaly or potential attack. This shift opened the door to leveraging sophisticated Artificial Intelligence (AI) and Machine Learning (ML) techniques.

2.2 Existing Systems: Traditional Methods and Challenges

Current approaches to network intrusion detection can be grouped into traditional signature NIDS, and early statistical or conventional machine learning-based anomaly systems. While both have contributed to security, they possess significant deficiencies in the context of modern cyber warfare.

Traditional NIDS Drawbacks:

The reliance on a constantly updated database of known attack signatures is both the strength and the Achilles' heel of signature-based systems. They are entirely ineffective against zero-day threats, as a signature takes time to create and deploy, leaving networks vulnerable in the interim. Furthermore, signature-based systems are susceptible to evasion techniques, where attackers subtly modify attack structures to bypass defined patterns. Practical NIDS deployments also face difficulty capturing all packets in high-traffic networks and struggle significantly to inspect encrypted traffic content, creating major blind spots as a large percentage of modern traffic is encrypted.

Traditional Machine Learning Drawbacks:

Early anomaly detection systems employed conventional ML algorithms such as Support Vector Machines (SVM), Random Forests (RF), and k-Nearest Neighbors (k-NN). Although these models represent an improvement over purely statistical methods, they suffer from two major limitations. First, they rely heavily on **handcrafted feature extraction** (feature engineering). This manual process of selecting relevant features and reducing data dimensionality is time-consuming and risks omitting subtle, yet critical, multi-dimensional patterns inherent in complex modern network attacks. Second, when implemented using supervised learning, these models require large volumes of labeled training data. Consequently, like signature-based systems, they falter severely when encountering novel (zero-day) attacks, as they lack prior knowledge of the threat patterns. This necessitates a model capable of automated feature learning and adaptation in dynamic, high-volume environments.

2.3 Proposed System Foundation: Deep Learning Architectures

Deep Learning (DL) provides a technological solution to the feature engineering and complexity challenges faced by traditional ML. DL models, by constructing Deep Neural Networks (DNNs), are capable of automatically learning and fitting highly complex, non-linear features directly from large volumes of raw or near-raw data, thus eliminating the time-consuming and error-prone process of manual feature extraction.

A range of DL models has been successfully applied to Intrusion Detection Systems (IDS), including Deep Belief Networks (DBNs), Multilayer Perceptrons (MLPs), Autoencoders, and Recurrent Neural Networks (RNNs). The choice of architecture depends heavily on the input data representation.

Necessity of Temporal Modeling

Network traffic inherently consists of sequential flows characterized by temporal dependencies, such as the time between subsequent packets, the sequence of port usage, and flow duration statistics. Therefore, models capable of handling time-series or sequential data are crucial for effective NIDS. Recurrent Neural Networks (RNNs) and their advanced variants are perfectly suited for this task.

Long Short-Term Memory (LSTM) networks are a refinement of RNNs, specifically designed to mitigate traditional RNN limitations like the vanishing or exploding gradient problems, allowing them to remember long-term dependencies in sequences. **Bidirectional LSTMs (BiDLSTM)** further enhance performance by processing the input sequence data both forward and backward, capturing contextual information from both preceding and succeeding flow events, leading to more accurate pattern detection. Studies confirm the efficacy of BiDLSTM-based IDS architectures in reducing false alarm rates and achieving high detection accuracy, particularly for subtle attacks.

Hybrid Superiority for Granularity

While LSTMs excel at temporal analysis, Convolutional Neural Networks (CNNs), which are often associated with image processing, can be leveraged to extract robust spatial features when network flow data is treated as a one- or two-dimensional matrix.

Research shows that **hybrid models**, which combine the strengths of different architectures, yield superior accuracy and detection granularity. Architectures combining CNNs (for automated spatial feature extraction) with LSTMs or Gated Recurrent Units (GRUs) (for temporal pattern analysis) can effectively capture both the static properties of a flow and its dynamic, time-based evolution. For example, the CNN-LSTM-GRU architecture has been demonstrated to achieve superior performance, nearing 100% accuracy in binary

classification tests on complex datasets, highlighting the benefit of fusing spatial and temporal feature extraction processes.

2.4 Advantages of Deep Learning in NIDS

The shift to deep learning provides profound operational advantages over previous technologies, fundamentally transforming the capability of intrusion detection.

Zero-Day Capability: Deep learning enables genuine anomaly detection by learning a precise baseline of normal behavior. Any significant deviation from this learned profile is flagged, regardless of whether a matching signature exists. This capability is crucial for identifying novel or zero-day attacks, which bypass conventional signature databases.

Reduced Operational Overhead and Scalability: By automating complex, high-dimensional feature extraction, DL models significantly reduce the manual effort and expertise previously required to tune and maintain traditional ML models. This automation enhances the system's scalability and responsiveness to changing network conditions.

Improved Accuracy and Sensitivity: DL models, especially advanced sequential or hybrid architectures, can process high-dimensional network data and model complex environmental factors with exceptional precision. This results in significantly improved detection accuracy and sensitivity, and, crucially, supports the goal of lowering the historically high false positive (FP) rate associated with earlier, less sophisticated anomaly systems. The capacity to process high-dimensional features from complex data streams, such as those extracted from IoT networks, makes DL the most robust solution for securing the expanding digital world.

Table 1 provides a high-level comparison of the technical merits of the major intrusion detection methodologies, justifying the proposed DL approach.

Table 1: Comparison of Intrusion Detection Methodologies

Feature	Signature-Based NIDS	Traditional ML Anomaly Detection	Deep Learning Anomaly Detection (Proposed)
Threat Detection Scope	Known attacks only (pattern matching)	Known/simple anomalies	Known, novel (zero-day), and subtle behavioral anomalies
Feature Engineering	N/A (Relies on predefined patterns)	Manual, time-consuming, sensitive to selection	Automatic feature learning via DL layers (e.g., CNN, Autoencoder)
Performance vs. Zero-Day	Fails entirely	Limited (supervised falters without labeled data)	Highly effective due to learned dynamic behavior baseline
Handling Encrypted Traffic	Limited content inspection capacity	Behavior/Flow analysis only	Behavior/Flow analysis (Temporal pattern recognition)

Chapter 3: Analysis and Requirements Specification (SRS)

3.1 Introduction: Operational Analysis and Context

The analysis phase is critical for bridging the gap between theoretical deep learning capability and the stringent requirements of a real-time, high-throughput Network Intrusion Detection System. A foundational operational reality dictates that in traditional NIDS processing, the detection phase is the single greatest computational consumer—requiring approximately 30 times the service time of packet decoding. Integrating complex deep learning models into this environment significantly increases the computational load, necessitating precise identification of hardware and software requirements to ensure low-latency operation.

The proposed system utilizes flow-based feature data, which encapsulates elementary network information such as Packet Size, Source/Destination Ports, and Time between subsequent packets. While these features are highly effective for modeling temporal anomalies using LSTMs, the process of flow reconstruction, feature calculation, and sequence generation introduces computational overhead. A thorough Software Requirements Specification (SRS) is essential to guide the implementation of a scalable, high-performance solution.

3.2 Software and Hardware Requirements Specifications (SRS)

The SRS defines the complete behavior and the operating environment of the developed DL-based NIDS, ensuring a clear understanding of the system's capabilities and operational constraints.

3.2.1 User Requirements

The primary users of this system are security analysts, network operators, and incident response teams. Their requirements center on speed, accuracy, and usability:

- **Real-time Visibility:** The system must continuously monitor network traffic and provide detection alerts immediately upon identification of an anomaly.
- **Actionable Output:** Prediction results must be displayed rapidly with a clear confidence score and sufficient ancillary information (e.g., source IP, flow metrics) to assist criminal investigations and forensic analysis.
- **Integration:** The system output must be easily consumable and integrate into existing security tools, particularly Security Information and Event Management (SIEM) systems, for centralized logging and coordinated response.
- **Low False Alarm Rate:** The system must achieve a high true positive rate while aggressively minimizing the rate of false positives (FPs) to prevent operator fatigue and maintain trust in the system's alerts.

3.2.2 Software Requirements

The software environment must be robust, flexible, and optimized for high-performance deep learning tasks.

1. **Operating System (OS):** Linux distributions (e.g., Ubuntu, CentOS) are recommended. These operating systems offer superior performance for network Input/Output (I/O) operations and kernel-level packet processing, which is critical for minimizing latency in the capture stage.
2. **Programming Language:** Python 3.8+ is the required language due to its strong community support and the mature ecosystem of advanced machine learning and data analysis libraries (e.g., NumPy, Pandas).
3. **Deep Learning Framework:** TensorFlow/Keras or PyTorch, with explicit and robust support for CUDA/GPU acceleration, is mandatory for efficient model training and high-speed inference.
4. **Data Capture/Flow Extraction Tools:** Mandatory integration of high-performance flow analysis tools such as CICFlowMeter or similar Libpcap-based utilities is required for extracting the 49 to 80 statistical flow features necessary for the DL model input.
5. **Database/Logging:** A time-series optimized database solution is required for the secure storage of raw flow logs, extracted features, and alerts, supporting high-speed write operations and efficient retrieval for forensic analysis.
6. **Web/Interface Frameworks:** Lightweight web frameworks such as Flask or Streamlit are suitable for developing the real-time visualization dashboard and user interface.

3.2.3 Hardware Requirements

Hardware selection is the most critical constraint in developing a real-time NIDS, as it must manage massive flow volume and the computationally expensive detection phase without dropping packets or introducing unacceptable latency.

Table 2: Hardware Requirements for Real-Time Deep Learning NIDS

Component	Minimum Specification	Justification for Real-Time Performance
Network Interface Card (NIC)	10 Gbps (Dedicated/Monitoring Mode)	Essential for high-throughput packet capture, preventing bottlenecks and drops when traffic exceeds 50 Mbps.

Component	Minimum Specification	Justification for Real-Time Performance
Graphics Processing Unit (GPU)	NVIDIA w/ CUDA (e.g., RTX series, 8GB+ VRAM)	Mandatory for accelerating deep learning inference, mitigating the 30x processing bottleneck of the detection phase.
Processor (CPU)	Intel Core i7 / AMD Ryzen 7 (High Clock Speed)	Required for rapid decoding, preprocessing, and managing the memory-intensive feature extraction and sequence generation processes.
RAM (Memory)	16 GB DDR4/DDR5 (Recommended 32 GB+)	Critical for storing large flow databases, caching data for the application, and model weights in memory, minimizing slow disk I/O latency.
Storage	512 GB SSD (Solid State Drive)	Necessary for rapid access to rule databases, model weights, and high-speed logging of security events.

The need for high-speed components is driven by the fact that the detection phase consumes the largest share of system resources. By utilizing a dedicated NVIDIA GPU with CUDA support, the computationally heavy mathematical operations of the DL model can be parallelized and executed rapidly, ensuring that the necessary complex behavioral analysis occurs within the tight timing constraints of high-speed network traffic. Dedicated, high-speed NICs (10 Gbps) are essential to prevent packet drops that occur when traffic exceeds typical 50 Mbps limits on non-optimized systems, thus ensuring complete visibility into potential intrusions.

3.3 Key Datasets and Feature Engineering

The training and validation of a robust anomaly detection system rely on contemporary, realistic network traffic datasets that accurately model both benign activity and modern attack scenarios.

Dataset Selection

The project focuses on contemporary benchmark datasets:

1. **UNSW-NB15:** This dataset was generated using the IXIA PerfectStorm tool to create a hybrid of real normal activities and synthetic contemporary attack behaviors. It includes 49 features derived using Argus and Bro-IDS tools, covering nine attack types, including Exploits, Fuzzers, Backdoors, and Generic attacks. It provides a comprehensive set of records, totaling over two million.
2. **CSE-CIC-IDS2018:** Designed to represent realistic modern network events, this dataset includes 80 features extracted using CICFlowMeter-V3 and covers seven modern attack scenarios, such as Brute-force, Heartbleed, Botnet, DoS, DDoS, and Web attacks. This dataset is valuable for training models on large-scale DDoS and Botnet activities, which are common financial motivations for hackers.

Feature Analysis and Preprocessing

Both selected datasets employ **flow-based features**, which characterize sequences of packets aggregated into network flows rather than inspecting individual packet payloads. This flow-based representation is crucial as it aligns perfectly with the proposed BiDLSTM architecture, allowing the model to detect anomalies in the temporal behavior (e.g., unusual sequence of port activity, inconsistent flow duration statistics).

Preprocessing is mandatory to handle inherent data challenges:

- **Handling Imbalance:** Network datasets exhibit severe class imbalance, where benign traffic vastly outweighs malicious samples (e.g., Benign traffic is over 83% of the CICIDS2018 dataset). Techniques like SMOTE (Synthetic Minority Oversampling Technique) or strategic undersampling of the majority class must be employed to ensure the model does not become biased towards predicting only the benign class.
- **Normalization and Scaling:** Features must be scaled (e.g., Min-Max normalization) to ensure convergence consistency during DL model training and to prevent features with larger magnitude values from dominating the learning process.

- **Sequence Generation:** For input into the LSTM model, the structured flow data must be transformed into time-series sequences using a fixed-size **sliding window** method. This process generates the input vectors, , required by the recurrent network to predict subsequent flow behavior, with deviations indicating anomalies.

3.4 Algorithms and Detection Workflow

The core logic of the system is the **Real-Time Anomaly Detection Algorithm using a BiDLSTM/CNN-LSTM Hybrid Model**. This algorithm outlines the logical steps necessary to achieve continuous, high-speed monitoring and classification.

Algorithm: Real-Time Anomaly Detection using BiDLSTM/CNN-LSTM Hybrid

1. **Start and Initialization:** The system initializes, loads the pre-trained DL model weights onto the GPU, and activates the Packet Capture Module to begin monitoring the network via the dedicated NIC.
2. **Continuous Data Capture:** The Packet Capture Module continuously sniffs raw network traffic at the kernel level and streams the packets to the Feature Processing Module.
3. **Flow Reconstruction and Feature Extraction:** The Feature Processing Module aggregates individual packets into bidirectional network flows. Relevant statistical flow features (e.g., Source/Destination Ports, flow duration, protocol flags) are calculated in real-time.
4. **Sequence Generation:** The continuous stream of flow features is transformed into fixed-length time-series sequences using a sliding window technique. This sequence is normalized, scaled, and prepared as the direct input () for the Deep Learning Detection Engine.
5. **Deep Learning Inference (GPU-Accelerated):** The sequence batch is fed through the GPU-accelerated BiDLSTM or Hybrid model. The model executes inference, utilizing its learned temporal patterns to calculate the likelihood of the flow sequence belonging to a benign class or an attack class.

6. **Anomaly Threshold Check:** The output probability distribution is checked against a predefined, empirically optimized anomaly threshold. If the probability of an attack class or the magnitude of reconstruction error (if using an Autoencoder component) exceeds this threshold, the event is flagged as an intrusion.
7. **Logging and Alerting:** A detailed, timestamped alert is immediately generated, logging all necessary ancillary information (source, destination, feature set) in the Database Service. The alert is simultaneously forwarded to the SIEM system.
8. **Continuous Monitoring:** The system loops back to the Data Capture stage (Step 2), ensuring zero-delay, continuous monitoring of the network environment.

Chapter 4: Modular System Design and Architecture

4.1 Introduction: Architectural Principles for Real-Time Systems

The design phase transforms the high-speed requirements identified during the analysis into a structured blueprint capable of physical implementation. For a real-time NIDS, the architecture must adhere to two paramount principles: **modularity** and **pipeline efficiency**. Modularity ensures that the various components—from packet sniffing to deep learning inference—are decoupled, allowing for individual optimization, upgrades (e.g., switching DL models), and robust maintenance without impacting the entire system. Pipeline efficiency ensures that the heavy computational load of the Detection Engine is managed optimally via GPU acceleration, preventing data backlogs and ensuring low latency, a primary concern when dealing with high-throughput network traffic.

The proposed design defines five essential, interconnected operational modules, structured in a linear pipeline to support the high data rate necessary for instantaneous threat analysis.

4.2 System Architecture (Modular NIDS Blueprint)

The proposed architecture is centered on a continuous data processing pipeline, where high-speed components handle the initial data ingestion and highly optimized DL models perform the complex analysis.

4.2.1 Input/Capture Module

This module represents the hardware-software interface. It utilizes the dedicated high-speed NIC (10 Gbps) to monitor network traffic non-intrusively (e.g., mirrored port mode). Its primary function is robust, kernel-level packet capture, minimizing computational overhead at this initial stage. The capture process must be highly efficient, minimizing packet drops and ensuring that all raw traffic is collected and buffered before being passed to the next stage.

4.2.2 Feature Processing Module

This critical intermediate module is responsible for transforming raw packet data into structured, sequential feature sets required by the DL model. This software layer performs:

- **Packet Decoding and Flow Reconstruction:** Aggregating individual packets into logical, bidirectional network flows based on connection characteristics (5-tuple keys).
- **Statistical Feature Calculation:** Calculating the 49 to 80 statistical flow features (e.g., packet counts, time distributions, flag status, byte ratios) using tools like CICFlowMeter.
- **Sequence Generation:** Applying the sliding window technique to flows, generating the time-series sequences that serve as the input for the recurrent network.
- **Normalization and Scaling:** Performing necessary preprocessing (Min-Max normalization) to prepare the batch data for optimal GPU processing efficiency.

4.2.3 Deep Learning Detection Engine

This is the core intellectual component of the NIDS. It houses the pre-trained, robust DL model (BiDLSTM or CNN-LSTM hybrid). The weights are loaded directly onto the dedicated GPU memory. The engine executes real-time inference on the batched sequences received

from the Feature Processing Module. By performing parallelized, low-latency calculations on the GPU, the engine determines the anomaly score or attack classification probability for the current network sequence. This stage overcomes the historical computational bottleneck associated with the detection phase.

4.2.4 Logging and Database Service

This module is essential for security compliance and post-incident forensics. It securely stores raw network flows, the extracted features used for detection, and all triggered alerts in a resilient, high-speed database. Maintaining detailed logs allows security teams to reconstruct attack timelines and confirm the nature of the detected anomaly.

4.2.5 Output and Alerting Module

This module provides the human-machine interface and integration point. It receives the classification results from the Detection Engine, validates them against the configured anomaly threshold, and generates immediate, prioritized alerts. Crucially, this module is designed to integrate seamlessly with external enterprise security systems (SIEM/SOAR) via standard protocols (e.g., API calls, syslog), ensuring the localized NIDS intelligence is transformed into an actionable, enterprise-wide incident response. It also includes a **Data Visualization Dashboard** for security analysts to monitor real-time network health and view historical attack trends.

4.3 Data Flow and Component Interaction

The system operates as a continuous data stream pipeline, with tightly integrated components working to minimize total detection latency.

The data flow begins with **Raw Traffic** being captured by the **Capture Module**. This traffic moves to the **Feature Processing Module**, where the CPU reconstructs flows and calculates features—this process represents a necessary sequential step that requires optimization to avoid becoming a bottleneck. Once transformed into sequential, time-series feature vectors, the data is pushed to the **Deep Learning Detection Engine**. This transition is critical: efficient buffering and batching must ensure the GPU is continuously utilized without waiting for CPU processing. The GPU performs the core inference, outputting an Anomaly Score in microseconds.

If the Anomaly Score exceeds the threshold, the **Alerting Module** is immediately triggered. This module simultaneously sends a detailed, timestamped **Alert** to the **SIEM System** for broader context correlation and coordinates the logging of the event in the **Database Service**. The use of this pipeline architecture, combined with GPU acceleration for the analysis-heavy Detection Engine, ensures the system maintains low latency while processing the high volumes required for real-time monitoring.

Chapter 5: References

1. Suk, H.-I., Lee, S.-W., & Shen, D. (2014). Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis. *NeuroImage, 101*, 569–582.
2. Payan, A., & Montana, G. (2015). Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks. *arXiv preprint arXiv:1502.02506*.
3. Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset. Available at: https://adni.loni.usc.edu.
4. Open Access Series of Imaging Studies (OASIS) dataset. Available at: https://www.oasisbrains.org.
5. SentinelOne. Vulnerabilities and Breach Statistics. 2024 Report.
6. Secureworks. Boardroom Cybersecurity Report 2024.
7. Fortinet. Evolving Cybersecurity Landscape: Top Cyber Threats in 2025.
8. Center for Strategic and International Studies (CSIS). Significant Cyber Incidents (February 2025 report).
9. Wikipedia. Anomaly-based intrusion detection system.
10. Sahoo, S., et al. (2020). Hybrid Anomaly Detection in Consumer Networks with Limited Resources. *MDPI Electronics, 9(6)*, 923.
11. Babiuch, M., et al. (2025). Taxonomy and Comparison of Intrusion Detection Systems. *arXiv preprint arXiv:2502.06688v2*.

12. General International Group. Advantages and Disadvantages of Intrusion Detection System (IDS) Types.

13. Corelight. Signature-based Detection.

14. Fidelis Security. Signature-Based vs. Anomaly-Based IDS.

15. Abro, S. A., et al. (2024). Comparison of Traditional vs. AI-Based Intrusion Detection and Prevention Systems: Efficiency and Accuracy. *ResearchGate*.

16. Al-Hajji, M., et al. (2024). A Hybrid Deep Learning Intrusion Detection System Based on Improved Feature Selection. *MDPI Applied Sciences, 15(4)*, 1903.

17. Ergun, O. (2024). AI vs. Traditional Methods in Intrusion Detection: A Comparative Analysis. *Orhan Ergun Network Engineering*.

18. Sun, M., et al. (2024). A Survey on Intrusion Detection Systems Using Machine Learning and Deep Learning Techniques. *MDPI Information, 16(7)*, 515.

19. Malla, S., et al. (2024). Deep Learning Algorithms Used in Intrusion Detection Systems—A Review. *arXiv preprint arXiv:2402.17020*.

20. Malla, S., et al. (2024). A Comprehensive Survey on Deep Reinforcement Learning for Network Intrusion Detection Systems. *arXiv preprint arXiv:2410.07612v1*.

21. Al-Mhiqani, M. S. et al. (2021). Real-Time Network Intrusion Detection Based on Deep Learning. *MDPI Sensors, 13(5)*, 111.

22. Aljameel, S.S., et al. (2022). Network Intrusion Detection System using Lion Optimization Feature Selection with a Deep Learning (NIDS-LOFSDL). *ResearchGate*.

23. Chudasma, P. (2020). Practical real-time intrusion detection using machine learning. *DBS eSource*.

24. Malla, S., et al. (2024). Deep Learning Algorithms Used in Intrusion Detection Systems—A Review. *arXiv preprint arXiv:2402.17020v1*.

25. Špitalský, D., & Babiuch, M. (2025). Taxonomy and Comparison of Intrusion Detection Systems. *arXiv preprint arXiv:2502.06688v3*.

26. Kaggle. Network Intrusion Detection (Dataset Snippet).

27. Ramos, S. (2024). Awesome-Cybersecurity-Datasets (GitHub).

28. U.S. DoD. Common Cybersecurity Acronyms.

29. SANS Institute. Glossary of Terms.

30. Cato Networks. 26 Cybersecurity Acronyms and Abbreviations.

31. Al-Mhiqani, M. S., et al. (2024). Deep Learning Algorithms Used in Intrusion Detection Systems. *arXiv preprint arXiv:2402.17020v1*.

32. Smith, J., et al. (2024). A Hybrid Anomaly-Based NIDS Integrating Deep Learning Algorithms. *PMC*.

33. YouTube. Intrusion Detection System using Deep Learning (LSTM) and Python.

34. Tencent Cloud. What are the hardware requirements for IDS?

35. Luo, W., et al. (2002). Optimizing intrusion detection systems for high-speed networks. *Georgia Tech GTISC*.

36. MITRE Corporation. Intrusion Detection: Lowering Requirements.

37. IBM. Intrusion Detection System.

38. Kim, J., et al. (2022). Deep learning model for one-class classification using convolutional neural networks and autoencoder for anomaly detection in network traffic. *PMC*.

39. Khan, A., et al. (2024). An Anomaly Detection Model with Feature Selection for Network Intrusion. *PMC*.

40. Al-Otaibi, M. N., et al. (2021). An Analysis of Feature Engineering and Machine Learning Techniques in NIDS Datasets. *arXiv preprint arXiv:2108.12722*.

41. Balarabe, T. (2023). Network Intrusion Detection with LSTM. *Medium*.

42. Junaid, M., & Alam, M. (2024). Network Intrusion Detection based on LSTM. *BC Publication*.

43. Ali, A. R., et al. (2025). Flow-Based Intrusion Detection in IoT Networks. *MDPI Sensors, 25(7)*, 2288.

44. Abas, Z. A. H., & Othman, R. M. (2023). Advanced Intrusion Detection System Based on Deep Learning. *MDPI Sensors, 24(24)*, 7924.

45. Harvi, I. (2024). NIDS: Real-Time Network Intrusion Detection (GitHub).

46. Fortinet. Intrusion Detection System.

47. IBM. IDS and SIEM (security information and event management).

48. Kumari, S. N., et al. (2020). Host-based Intrusion Detection System using Hybrid Machine Learning. *IJCRT*.

49. Hasan, R., et al. (2022). General architecture of deep learning-based IDS. *ResearchGate*.

50. Alzahrani, A., et al. (2023). A Deep Learning-Based Classification Model for Handling Large Data Volumes. *MDPI Electronics, 12(19)*, 4050.

51. IBM Developer. Machine learning and deep learning architectures.

52. Qureshi, N. U. (2020). Proposed modular deep neural network architecture. *ResearchGate*.

53. Popovic, J., et al. (2023). Modular Network Intrusion Detection Architecture Capable of Simulating Cyberattacks. *X-Mol*.

54. Alkhalil, E., et al. (2023). An Enhanced Hybrid Deep Learning Model for Intrusion Detection in IoT and IIoT. *MDPI Applied Sciences, 13(9)*, 222.

55. Fortinet. What Is An Intrusion In Cybersecurity?

56. IBM. Anomaly-based detection methods use machine learning.

57. Wikipedia. Intrusion detection system.

58. Koorsen. Best Practices for Deploying an Intrusion Detection System.

59. Canadian Institute for Cybersecurity (CIC). CIC-IDS2018 Dataset.

60. Kalkan, F., et al. (2018). Dataset Analysis of CICIDS2017 and CSE-CIC-IDS2018.

61. Fenzl, M., et al. (2024). LycoS-IDS2017 to the well-known CSE-CIC-IDS2018 dataset. *arXiv preprint arXiv:2402.10974v1*.

62. Heinze, F. (2021). CSE-CIC-IDS2018 traffic distribution.

63. Wells, D. (2022). UNSW-NB15 dataset. *Kaggle*.

64. Research Data Australia. The UNSW-NB15 dataset.

65. Canadian Institute for Cybersecurity (CIC). CIC UNSW-NB15 Augmented Dataset.