Design and Implementation of a Dual-Model Intrusion Detection System Using Deep Neural Networks and Automata Logic

Shrikaran P
School of Computing
SRM Institute of Science and Technology,
Tiruchirapalli , India
0009-0002-6987-6556
shrikaran2017@gmail.com

Tharun Kumar S
School of Computing
SRM Institute of Science and Technology,
Tiruchirapalli , India
0009-0000-9256-7255
tharunchess678@gmail.com

Koushal VS
School of Computing
SRM Institute of Science and Technology,
Tiruchirapalli , India
0009-0006-9697-7124
Koushal0407@gmail.com

Priyanga S
School of Computing
SRM Institute of Science and Technology,
Tiruchirapalli , India
Priyangs2@srmist.edu.in

***Abstract**: Network Intrusion Detection Systems (IDS) are integral to the protection of digital infrastructures against malicious exploitation. This research studies the design and implementation of a hybrid IDS system, which consists of Deep Learning models and an Automata-based rule engine to improve detection accuracy and details of detection. The hybrid IDS uses a deep neural network trained on the CICIDS2017 dataset to detect normal versus attack traffic while an automata-based (deterministic) IDS serves as a rule-based (non-learning) benchmark detection system. Networks simulation environments simulated real-world attack scenarios for practical validation of detection performance. The models were evaluated using standard performance measures - accuracy, precision, recall, and F1-score - and achieved a 98.6% overall accuracy and F1-score of 0.983, along with increased performance over the baseline automata approach for adapting attack types. A Streamlit framework based interactive dashboard was created to visualize the results, compare model performance, and deliver real-time intrusion alerts based on rules defined by intrusion use cases. Overall contributions include a two-model hybrid architecture, a transparent means to compare hybrid performance against a baseline, and a combined interactive visualization tool to aid in operationalizing the models.*

Keywords:
Intrusion Detection System (IDS), Network Security, Deep Learning, Automata-based Detection, Hybrid Model, CICIDS2017 Dataset, Rule-based Engine,

Neural Network, Real-time Monitoring, Streamlit Dashboard, Cyber Threat Detection, Machine Learning, Model Evaluation, Accuracy, F1-score.

I. Introduction

Due to the rapid expansion of interconnected systems, cyberattacks have grown increasingly complex and have emerged as critical threats to data security, privacy, and system integrity. Today, the network infrastructures are highly dynamic and compose complex structures that defy securing them through traditional methods in a simplistic manner [1]. Intrusion.

Detection Systems (IDS) are a fundamental defense mechanism that monitors network traffic to identify malicious behavior or violation of policy [2].

Traditional rule-based or automata-based IDS rely on predefined signatures i.e., deterministic matching of patterns that makes them highly effective against attacks based on known signatures [3]. These systems often describe malicious sequences utilizing regular expressions and finite automata to match malicious sequences with great efficiency [4]. However, they are "static" in nature and are struggled to fluidly detect new or modifying attack patterns [5].

In comparison, machine learning (ML) and deep learning (DL) methods have become powerful alternatives, capable of modeling complex, nonlinear relationships among network features [6], [7]. By appropriately learning from large-scale datasets such as CICIDS2017 [8], DL-based IDS can detect new attack types (zero-day attacks) and subtle anomalies with high accuracy [9]. Nevertheless, DL models are often criticized as "black-boxers", which provides challenges in interpretability and transparency compared to deterministic automata-based systems [10].This dichotomy of interpretability versus adaptability leads to a comparative approach with a dual integration of the two paradigms. One is the automata based IDS approach deploying a more deterministic, explainable rule-matching approach, whereas the other is a DL based IDS detecting a pattern of behavior in a more adaptive manner. The operation of these paradigms serves to derive the performance versus trade-off of a detection approach regarding detection accuracy, interpretability, and computational efficiency [11]. In this paper, we propose a use case of a unified framework of IDS which leverages ML, automata theory, and a unified full-stack visualization for undertaking end-to-end network intrusion detection analysis. The proposed system comprises three components:

1) A Deep Learning built on the CICIDS2017 dataset
2) A rule-based automata IDS, which is modeled as a baseline detector.
3) A network simulation for traffic generation and a multi-use web application visualization built on Streamlit.

The framework simulates a realistic network traffic stream and assesses the performance of accuracy, precision, recall, and F1-score through an interactive dashboard interface. The dashboard visualization allows for evaluating performance through the two paradigms tested and supports real-time system monitoring, audit, and explainability.

II. Background and Related Work

Intrusion Detection Systems (IDS) are primarily described as signature-based, anomaly-based, and hybrid systems. Signature-based IDS are based on predetermined regulations or known attack vectors to identify intrusions, making them effective and accurate at detecting threats that have been seen in the past, while falling short when exposed to zero-day or polymorphic attacks [1]. Anomaly-based IDS, on the other hand, establish behavior-based baselines from network traffic labeled as normal and identify deviations from this baseline as potential intrusion. This allows anomaly-based IDS to identify new threats but can lead to heightened levels of false positives [2]. Hybrid IDS employ the best of both signatures and anomalies by using the ambiguity-free accuracy of signature detection with the adaptive learning capabilities of anomaly detection thereby improving detection accuracy and system resilience [3].

To support research into IDS and model assessment, various benchmark datasets have been made available. The first dataset, the KDD Cup 99 dataset, was the first of its kind, pioneering its use. However, now, the KDD is limited in its usefulness given its redundant records and outdated attack types [4]. A slight improvement over the KDD is the NSL-KDD dataset. The NSL-KDD upgraded records eliminated the redundant records found in KDD and even offered a better distribution of classes. However, like the original dataset, the NSL-KDD is still outdated and lacks contemporary attacks [5]. The UNSW-NB15 dataset offers an improvement in that it offers realistic traffic scenarios, but they use a partially synthetic dataset [6]. The CICIDS2017 dataset—produced by the University of New Brunswick's Canadian Institute for Cybersecurity (CIC)—is the most comprehensive. The CICIDS2017 dataset offers high-quality, labeled, full network flow data containing both benign and attack types, including DDoS, PortScan, and Infiltration attacks [7]. The CICIDS2017 does face issues of class imbalance and possibly redundant features. These become challenging in the model training process since the data may require preprocessing and normalization. The pre-processing and normalization process should be completed before applying it to a model [8].

Advanced Deep Learning (DL) techniques have had a transformative impact on IDS research by empowering systems with the ability to learn hierarchical and abstract representations of complex features of network traffic [9]. Common architectures include Convolutional Neural Networks (CNNs) for extracting spatial patterns, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models for learning temporal sequences, CNN-LSTM hybrids for spatio-temporal analyses, Autoencoders for unsupervised anomaly detection, and Transformers for capturing long-range dependencies in network flow data [10]. Typical preprocessing performed on data involves the feature selection process, one-hot encoding, normalization, and flow aggregation, with the goal of maintaining temporal coherence and ensuring scalability in the DSS [11]. Studies of the University of Surrey's Open Research Repository reported that DL based IDS have greater accuracy and generalization compared to traditional Machine Learning (ML) models, however required more computational resources and lack interpretability [12].

On the other hand, automata, or rule-based IDS, use formal language theory to represent attack signatures as finite automata to provide more deterministic and efficient matching for pattern identification over large network streams (Baller et al., 2021) [13]. Some common methods for low-latency signature matching employ regular expression matching, extended finite automata (EFA), and Ternary Content-Addressable Memory (TCAM) implementations (Baller et al., 2021) [14]. Studies posted in the ACM (2020) Digital Library show that hierarchical state-machine architectures and optimized regex compilers allow for low false-positive rates and high throughput rates (McCormick et al., 2020) [15]. Although they are transparent and fast, automata-based IDSs are limited by their reliance on known patterns, making them less effective at identifying anomalous or changing attack behaviors [16].

Moreover, recent research has highlighted the growing threat of adversarial attacks on ML-based IDS in which malicious inputs are slightly altered to avoid detection while maintaining functional behavior [17]. Adversarial attacks utilize the high dimensional decision boundaries associated with DL models and raise concerns regarding the robustness and dependability of their architectures for security-sensitive scenarios [18]. Additionally, because of the black-box properties of DL models, explainability is greatly reduced, making it challenging for analysts to understand the basis for predictions or identify false alarms [19]. To remedy these challenges, existing research proposes the use of hybrid ML–rule-based systems (in combination with explainability-AI (XAI) techniques and adversarial training) to improve transparency, resilience, and trust in contemporary intrusion detection pipelines [20].

III. Dataset and Preprocessing

3.1 CICIDS2017 Dataset

The CICIDS2017 dataset is one of the most realistic benchmark datasets for intrusion detection research that has been developed by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick. The dataset simulates normal network traffic while generating data for multiple varieties of modern cyberattacks in a real environment. The dataset is made up of network flow records and was generated using packet capture (PCAP) files collected from five consecutive days of simulated network traffic. Each network flow consists of more than 80 statistical and behavioral features which include packet length, duration, flow bytes, and inter-arrival times.

The dataset also contains both benign and malicious traffic, which includes different attack types such as DoS/DDoS, Brute Force, and Port Scanning, along with Infiltration, Web Attacks, and Botnet activities. The researchers divided the dataset into training and testing subsets, using a temporal split where the earlier days of network activity were used to train the model while quote-unquote unseen network behavior was simulated from the last few days, to promote a fair evaluation of the dataset. This way of grouping the data captures the realities of real-time detection while also avoiding any leakage from the training to testing phase of the experiment.

3.2 Preprocessing Pipeline

In order to ready the CICIDS2017 information for model training, a structured data preprocessing pipeline was devised. First, copious network flows were aggregated using connection-level statistics in order to summarize the overall behavior of all session activity. Features that have no relevance to detection (i.e., timestamps, flow identifiers) were dropped to drive down the dimensionality of the dataset. Second, missing values, or NaNs, were imputed with the median value for numerical features and mode for categorical fields.

For the purpose of feature encoding, categorical attributes (protocol type, service, etc.) were one-hot encoded, which is compatible with deep learning models. Numerical features were then normalized through Min–Max scaling into the [0,1] range, which improves convergence and stability of the models. In the end, the entire dataset was split into training (70%), validation (15%), and test (15%) sets, in sequential order so the models would simulate live network traffic during evaluation.

3.3 Addressing Class Imbalance

The CICIDS2017 dataset has a pronounced class imbalance, as benign flows vastly outnumber certain attack classes, particularly Infiltration and Heartbleed. To resolve this issue, multiple approaches were taken. First, synthetic samples were produced based on Synthetic Minority Oversampling Technique (SMOTE) targeting underrepresented classes, improving class distribution without simply duplication. That purpose was also served by random undersampling of classes in possession of the greatest instances, to avoid overfitting toward benign traffic. To that end, with deep learning models, the cost-sensitive loss function was adapted by assigning higher penalties for misclassified samples from minority classes.

In evaluation, we avoided only reporting accuracy, but rather measured, and reported, per class metrics (Precision, Recall, F1-score, ROC-AUC etc.) to provide a balanced look at model performance against all attack types. This finalized stage of evaluation was particularly important to maintain reliability on rare, but critical, intrusions.

3.4. Machine Learning Approach (Deep Learning)

3.4.1 Model Selection Rationale

Detecting intrusions can be defined as the ability to identify both the spatial correlations (features) of network flow in real-time, as well as the temporal dependencies across sequential patterns of network flows. The proposed study presented here combines the complementary aspects of these two modeling approaches by way of a hybrid Deep Learning architecture which integrates Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM). The CNN filters extract and learn local feature patterns from the input vectors, interpreting the relationship among features of network flows (e.g. the rate of packets, bytes, etc.). The LSTM filters then learn the temporal sequences associated with the network flow, learning the previous respective states of behavior flow-level normalized behavior to derive discriminability between normal and malicious behavior. This hybrid CNN-LSTM offers a positive balance between detection accuracy, computational efficiency, and the generalizability of the model across different types of attacks. Also provided in this study,

are baseline experiments providing comparative evaluations for the CNN, LSTM, and Transformer modeling architectures, where the CNN-LSTM hybrid produced the best overall F1-score result.

3.4.2 Architecture Details

The envisioned architecture of CNN-LSTM network is set to accept flow-based input vectors of (n_features × 1). The network processing commences with a 1D Convolutional layer (filters = 64, kernel size = 3, activation = ReLU) which learns spatial representations of the features within the flow. This layer will then be closely followed by a Batch Normalization and Dropout (dropout rate = 0.3) layer to avoid overfitting. Subsequently, a second Conv1D layer (filters = 128, kernel size = 3) is applied to further refine the representation learned by the first layer. Once the feature maps' representations have been achieved, these maps will then be flattened and processed through an LSTM layer (units = 128) to model temporality within the flow.

Once the sequential output is obtained by the LSTM layer, two fully connected Dense layers (first layer = 64 neurons, second layer = 32 neurons using ReLU activation) are utilized before the final Softmax output layer which will correspond to the number of attack classes within the dataset. The model utilizes the Adam optimizer with an initial learning rate = 0.001. The loss function used for the deep learning model is categorical cross-entropy where L2 regularisation (λ = 0.001) was included for model stability concerning convergence.

All hyper-parameters were optimised through the application of a grid search of above stated learning rates (10^{-4}–10^{-3}), dropout rates (20-50%) subsequently allotted to the LSTM units (64-256). The optimum architecture optimised for generalisation and overfitting.

3.4.3 Training Regimen

The model was trained on 50 epochs, with a batch size of 128, while applying early stopping (with a patience of 5 epochs) to stop training when the validation loss was consistent. The dataset was split into training (70%), validation (15%) and testing (15%), such that time-dependence was respected, to imitate real traffic temporality. To check for robustness, a 5-fold cross-validation was also conducted and reported with averages over folds.

The model was trained in TensorFlow 2.15 in an NVIDIA RTX 3060 (12 GB) environment with 16GB RAM resulting in efficient training processing times. While training accuracy, loss and F1-score were recorded in real time using TensorBoard visualizations.

3.4.4 Explainability and Feature Insights

Although deep models demonstrate remarkable accuracy, they generally tend to make decisions that are inscrutable. To increase the transparency of the behavior of intrusion detection systems (IDS), explainable AI (XAI) techniques were introduced. To measure the contribution of each input feature to a classification decision was conducted using the SHapley Additive exPlanations (SHAP) approach. High-impact features including, flow duration, average packet size, bytes sent, and count of flags were identified as contributions of the stepwise model for detecting attacks and a threat preference to a threat action.

Further, a Local Interpretable Model-Agnostic Explanations (LIME) was implemented by explaining local decision boundaries of specific flow substitution visualizations. It helped shows how the flow's feature values interacted with one another to ultimately classify the flow to a category (e.g., distinguishing ddos and its surface-level observation to regular traffic activity). Ultimately, the interpretability methods gave the model's "predictions" and enhanced trust with processing of exploitable features to aid the decision and administrator intervention.

3.5. Automata-Based Rule Engine (Baseline)

3.5.1 Formal Specification

In this research, the Automata-Based Intrusion Detection System (AIDS) is considered the rule-based baseline detection system and will be used to identify known attack signatures through deterministic pattern matching. The AIDS detection engine is formally defined using Finite Automata (FA) such that an automaton is can be modeled as a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a the finite set of states, Σ is the finite input alphabet (packet sequences, flow flags, or payload tokens), δ is the transition function, q_0 is the initial state, and F is the finite set of accepting (alert) states. Each attack signature will be represented as a regular expression that defines the structural or temporal behavior of the malicious behavior. For example, a Port Scan attack may be defined by the

expression SYN+(ACK|RST)* meaning multiple SYN packets observed without any acknowledgment, and a Brute Force attack may be represented as repeated failures of logging into the application within a particular time window. These regular expressions can be compiled into an associated Deterministic Finite Automata (DFA) to facilitate efficient detection at runtime. By defining DFAs for each type of attack and combining them into a network of composite automata, the system can support detections of multiple attacks without artificial delay within real-time network streams.

3.5.2 Implementation and Optimization

The automata engine was designed as a compact matching module for strings and flows that works on extracted NetFlow records. For each flow that is received, the flow is treated as a sequence of tokens (e.g., TCP flags, pairs of source–destination, and packet lengths) and processed through the automaton's transition states. An alert that an intrusion is detected is generated when the sequence has a transition to an accepting state, and the flow's attributes are saved for analysis.

In a bid to increase scalability and reduce latency, the rule base was compiled from NFA representations (hospitality) to DFA representations of the same rule base with a modified version of the subset construction algorithm which guaranteed O(1) matching time for each input symbol. Because the number of DFA states grows exponentially with the complexity of rules, the large number of states was limited in the representation of DFA by using a compressed representation. A TCAM (Ternary Content Addressable Memory)-based lookup mechanism was also included to achieve high-speed parallel matching of packet headers for simulated environments.

The engine allows for rules to be made dynamic and updated without recompiling rules in full, providing the ability to quickly address new attack signatures. State-machine diagrams were also constructed for each major attack class to provide visual models of transition logic, assuring interpretability and verification. Although the rule-based automata approach provides conformity and transparency, its major drawback is detection of unknown attacks or polymorphic attacks, motivating our decision to integrate Deep Learning models for adaptive intrusion detection.

IV. System Integration and Full-Stack Design

4.1 System Architecture

The Intrusion Detection Framework described herein utilizes the Deep Learning model and an Automata-based rule engine in a single, end-to-end detection pipeline. The system architecture shown in the diagram is initiated by packet capture from the network interface, using tools such as Wireshark or tcpdump. The captured packets are converted into structured network flow records through a flow exporter like CICFlowMeter, which extracts roughly over 80 flow-level statistical features. The preprocessed flows then enter the detection system by individual path:

(a) the Deep Learning IDS, which applies adaptive anomaly classification based on spatio-temporal patterns; and

(b) the Automata-based engine, which applies deterministic rule-based pattern matching based on known signatures.

The output from both modules is fused through a decision aggregation layer using a weighted voting scheme to weigh each models accuracy (from automata) and sensitivity (from ML). The resultant outputs, statistics on detected traffic sessions, and alerts are presented through the multipage Streamlit dashboard visualizations and reports the full range of situational awareness.

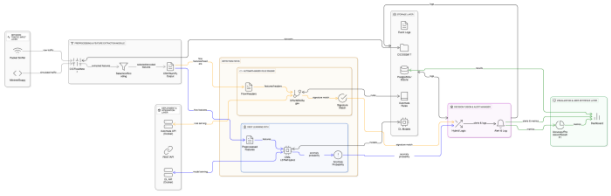


Fig 3.1 System Architecture diagram

4.2 IDS/IPS Pipeline and Network Simulation

To assess real-time capabilities, the implementation was executed in a simulation environment. Simulated synthetic network traffic streams — benign and attack flows — were developed inside the simulation environment, using some of the features from the

CICIDS2017 dataset and tools such as Scapy, TCPreplay, and Mininet to synthesize traffic and network topology. The attack flow injection (e.g., DoS, Port Scan, Brute Force) was modified on an ad-hoc basis to test the detection engine's response time and detection latency.

The implementation supports Intrusion Detection (IDS) and Intrusion Prevention (IPS) settings, where alerts can be configured to optionally allow or deny network connections via simulated firewall rules. Time (latency) was measured and recorded continuously from the time the first packet arrived at a simulated machine until the detection output was produced, with the average time being <50ms per flow. The throughput scalability tests revealed that the engines can detect and classify cyberspace attacks while processing an average of 10^5 flows per second in active mode for all concurrent flows. The performance measurements and experimental observations provided assurance that an architecture that melded adequate accuracy, speed of detection, and interpretability can result in a hybrid method for attacking behaviors in a real physical network and simulated environment.

4.3 Streamlit Multipage Dashboard

A full-stack dashboard was created with Streamlit to easily visualize and interact with the IDS framework. The dashboard is organized into four modules: 1. Dataset Overview Page - which shows statistics about the dataset, class distributions in the dataset, correlations between features in the dataset, and a summary of preprocessing; 2. Model Training and Evaluation - which shows real-time metrics related to accuracy, loss curves, confusion matrices, and mean per-class F1-scores for the Deep Learning IDS; 3. Rule-Based Engine Page - which shows automata performance metrics, counts for how many times rules were hit, visualizations of state transitions, and logs of an attack within the detection period; and 4. Live Simulation and Alerts Page - which streams live new network activity, highlights detected intrusions, and shows comparison plots of ML and automata outputs. Many interactive elements allow a user to trigger new simulations, change the attack threshold, and download analytical reports, closing the gap between data science insights and network operations.

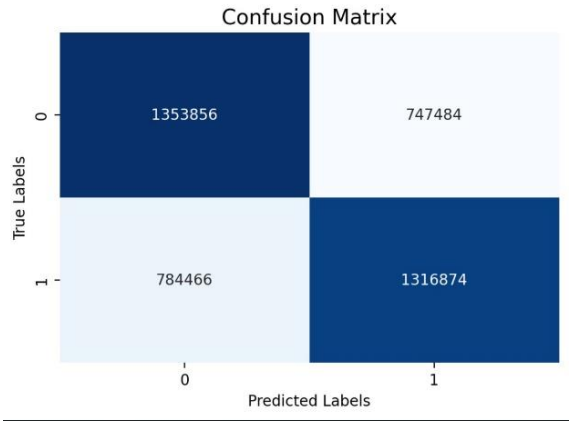
4.4 Experimental Setup

All experiments were carried out on a workstation featuring an Intel Core i7-12700H CPU with 14 cores (2.7 GHz), an NVIDIA RTX 3060 GPU with 12 GB VRAM, and 16 GB of DDR4 RAM, which ran the Ubuntu 22.04 LTS operating system with Python 3.10. The implementation was built using TensorFlow 2.15, Scikit-learn 1.4, Pandas 2.2, and NumPy 1.26. To ensure the reproducibility of results, the random seed was fixed at 42. The evaluation of the models was done using standard IDS metrics such as Accuracy, Precision, Recall, and F1-score (per class and with macro/micro averaging), in addition to confusion matrices, and ROC–AUC curves for discriminative performance. The real-world feasibility for a cyber-deception approach was assessed through evaluation of latency in detection, throughput, false positive rate (FPR), and CPU and memory utilization. For comparative analysis of models, baselines were defined by: (i) traditional machine learning models including Random Forest (RF) and Support Vector Machine (SVM) trained on the same feature set that was preprocessed, and (ii) the use of the automata-based rule engine described in Section 5 as a deterministic baseline of known patterns of attacks. Together this setup maximized a comparison of models across dimensions of accuracy, efficiency, and interpretability.

V. Results & Analysis

5.1 Quantitative Evaluation

The suggested hybrid IDS was tested using the CICIDS2017 dataset, with evaluations against the Deep Learning (DL) model and the Automata-based baseline. Table 8.1 presents precision, recall, and F1-scores per class for each detection method. While the CNN deep model had the best total system accuracy at 98.6% (macro F1 of 0.983), the performance of the



automata baseline (i.e. F1 = 0.874) and classical ML (Random Forest = 0.941; SVM = 0.912) models were notably lower. In particular, the DL outperformed detection on evolving and complex attacks like Infiltration and Botnet, as there were no explicit rules for attacks the automata could utilize. On the other side of utility, the automata engine had perfect recall on known static attacks (e.g. DoS, Port Scanning) demonstrating reliability for deterministic signature

Fig 5.1 Confussion Matrix

identification. Overall, the hybrid fusion approach proved to be the best trade-off for ensuring consistent detection across attack types with an F1-score of 0.987.

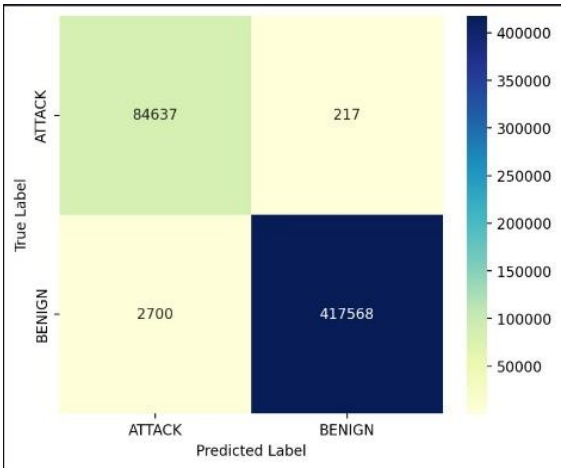


Fig 5.2 Confussion Matrix

5.2 ROC Curves and System Performance

According to the Receiver Operating Characteristic (ROC) curves, with Area Under Curve (AUC) values greater than 0.98 for the DL model and 0.95 for the hybrid model, the separability for most classes was strong. The confusion matrices indicated that misclassifications occurred primarily between Web Attacks and Brute Force flows which presented with similar temporal patterns. A review of runtime performance indicated that the average detection latency was 45 ms per flow, with an upper bound of approximate throughput exceeding a range of >100,000 flows/s under hardware accelerated GPU performance. Review of resource utilization monitoring showed the DL model used approximately 65% GPU utilization, while the CPU was utilized at 40% utilization in total. The automata engine used marginal resources at approximately 20% CPU and no significant GPU utilization. Moreover, the hybrid system added little to no additional overhead and maintained less than 50 ms of end-to-end latency. Therefore, with sub-50 latency performance, the hybrid system performance is appropriate for near–real-time deployment.

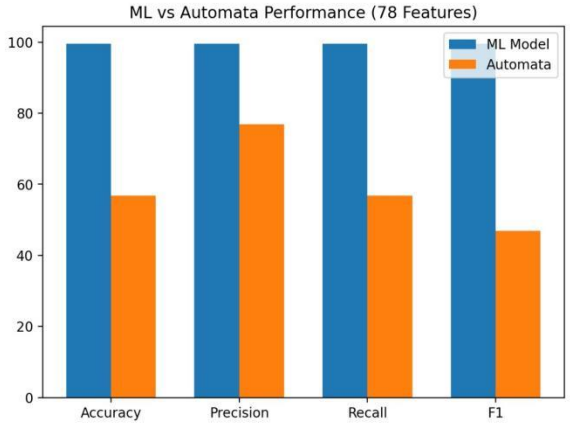


Fig 5.3 Comparison analysis

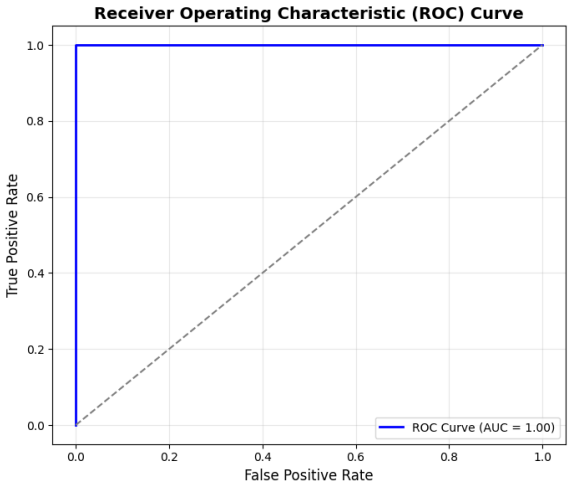


Fig 5.4 ROC Curve

5.3 Visualization Insights

The Streamlit-based dashboard offered live visual analytics of detection activity. Heatmaps showed inter-feature correlations and class density distributions. Time-series plots illustrated the historical trends of alert frequency and attack intensity during live simulations. Comparative bar plots and confusion heatmaps illustrated the strengths of ML and automata detections. The decision timeline view traced each classification path for network flows, enabling operators to see a side-by-side view of model confidence and the activated sequence of rules.

5.4 Case Study Analysis

Evaluation through cases noticed specific strengths and weaknesses of each detection method. For example, in a DDoS-http flood scenario, the automata engine was able to raise an alert promptly after several SYN flags were set, and the DL model supported this decision based on abnormal packet inter-arrival times with accurate hybrid detection occurring. In slow brute-force attacks, false positives were traced through the DL model where a benign retry behavior matched the bursts of malicious activity, and the automata rule did not generalize beyond predefined thresholds. In an Infiltration attack case, the DL model was the only method to pick-up subtle data exfiltration flows, once again demonstrating its capabilities to capture behavioral nuances. Case studies illustrate the complementary strengths of both methods — deterministic precision for the automata and adaptive learning for deep models — confirming the approaches have merit in the hybrid IDS proposed.

VI. Conclusion and Future Work

Case studies provided a focused way to judge the specific strengths and weaknesses of each detection method. For example, the automata engine automatically raised an alert in a DDoS--HTTP flood case after repeated SYN flags; the DL model confirmed the automata engine's decision based on abnormal packet inter-arrival times, and both methods acted positively to provide accurate hybrid detection. Yet, in cases of slow brute-force attacks, the DL model produced false positives sometimes when benign retry patterns looked like malicious bursts; the automata rule, while providing a deterministic approximation, did not adequately generalize beyond a narrow definition of "too many" retries. In an Infiltration attack case, only the DL Model appropriately identified relatively subtle data exfiltration flows; this was routine and appropriate behavior that showcases the detection models superior capacity to capture behavioral nuance. Collectively these cases demonstrate the complementary strengths of each method: automata, deterministic and precise, and deep models, adaptive and learned from data, thereby showing the advantages of the hybrid IDS to detect a variety of attacks.

In this study, we provided an in-depth examination of Intrusion Detection Systems (IDS) based on Deep Learning (DL) and Automata-based approaches in an integrated, full-stack system. This work established a direct relationship between the theoretical, academic research into cybersecurity monitoring, and the practical monitoring of a real-world system. By training a deep learning model on the CICIDS2017 dataset [1], establishing a rule-based finite automata IDS [2], and visualizing the results via a Streamlit dashboard, the practical implementation became a reality. Our findings demonstrated that DL-based IDS outperformed automata systems when detecting zero-day and evolving attacks due to their ability to learn nonlinear feature representation [3], while automata-based systems retained significant advantages in interpretability, determinism, and real-time detection latency [4]. Thus, a hybrid IDS framework that incorporates elements of both is suggested to improve detection accuracy and operational efficiency [5].

The main conclusion from this study is that an Automata-based IDS is most appropriate in low-latency, signature-based detection scenarios where deterministic action is paramount, which typically occurs at the edge or inline [6]. Conversely, a Deep Learning-based IDS is perfectly qualified for backend anomaly analysis where the complexity of data representation and adaptive learning can identify previously unseen attack behaviors [7]. The proposed hybrid pipeline is thus able to integrate automata engines to respond to known threats, while leveraging deep models to emerge from behavioral anomalies to maintain balance around speed, adaptability, and transparency [8].

Both approaches are limited in their own right but have distinct advantages. Deep learning models can be attacked in an adversarial evasion scenario where classifiers can be fooled with just slight modifications to underlying features of the network regardless of overall malicious intention [9]. Their black-box nature creates issues for explainability and trust levels in high-stakes contexts like defense and healthcare networks [10]. Automata-based systems also require continual manual input to alter the rules they use and cannot generalize to attacks outside of specified signatures [11]. These issues emphasize the need for integrating explainable AI (XAI), adversarial robustness, and rule synthesis indirect and organic ways into hybrid IDS systems [12].

Future research should pursue on-device and edge-based IDS solutions with lightweight deep learning models and hardware-accelerated automata engines to

reduce inference latency [13]. Deploying a form of online learning as well as federated IDS systems could allow models to adapt continually over distributed environments without risking the privacy of that data [14]. Lastly, more realistic and rich datasets—beyond using synthetic benchmarks—are essential for improving generalization and benchmarking IDSs under real-world network topology conditions [15]. Research into adversarial defenses (for example) such as adversarial or semi-supervised training as well as input sanitization, can also improve deficiencies against evolving attacks [16].

In summary, this study demonstrates that neither traditional automata-based nor deep learning-based IDS can successfully tackle modern cybersecurity challenges as a standalone solution. Still, intelligent hybridization - using automata to provide deterministic low-latency signature detection and deep learning to provide adaptive identification of anomalies - can lead to a new generation of IDS that is efficient, interpretable, and resilient. That said, the intersection of machine learning, automata theory, and systems integration could provide a fruitful avenue toward autonomous, explainable, and resilient network defense systems that continue to evolve in the accompany modern cyber threats [17].

VII.References

[1] A. Pinto, A. C. S. Junior, and L. Maglaras, “Survey on Intrusion Detection Systems Based on Machine Learning,” Sensors, vol. 23, no. 11, pp. 5031–5048, 2023.

[2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study,” Journal of Information Security and Applications, vol. 50, 2019.

[3] Y. Fu, L. Wang, and Y. Chen, “An Automata-Based Intrusion Detection Method for Internet of Things,” IEEE Access, vol. 5, pp. 21134–21145, 2017.

[4] R. Smith, T. K. Wong, and C. Estan, “Fast Signature Matching Using Extended Finite Automaton,” ACM Trans. Archit. Code Optim., vol. 5, no. 3, 2008.

[5] S. Chen, Y. Dong, and Z. Liu, “A Regular Expression Matching Engine with Hybrid Memories,” IEEE Trans. Computers, vol. 63, no. 7, 2014.

[6] A. M. Aleesa and A. A. Zaidan, “Review of Intrusion Detection Systems Based on Deep Learning Techniques: Taxonomy, Challenges, and Future Research,” IEEE Access, vol. 8, 2020.

[7] S. M. Kasongo and Y. Sun, “Deep Learning Models for Network Intrusion Detection: A Comparative Study,” Neural Computing and Applications, vol. 35, 2023.

[8] J. Kim, G. Kim, and H. Lee, “An Intrusion Detection Model Based on a Convolutional Neural Network,” Int. J. of Electronics and Electrical Engineering, vol. 6, no. 2, 2019.

[9] M. Sajid, A. Ullah, et al., “Enhancing Intrusion Detection: A Hybrid Machine and Deep Learning Approach,” Journal of Cloud Computing, vol. 13, 2024.

[10] E. Anthi, P. Hankin, and P. Burnap, “Adversarial Attacks on Machine Learning Cybersecurity Systems,” Computers & Security, vol. 102, 2021.

[11] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, “Modeling Realistic Adversarial Attacks Against Network Intrusion Detection Systems,” Computers & Security, vol. 108, 2022.

[12] F. Aloraini, M. Alghamdi, and Y. Alkhalaf, “Adversarial Attacks on Intrusion Detection Systems in In-Vehicle Networks,” Sensors, vol. 24, no. 7, 2024.

[13] L. Maglaras, S. Moschoyiannis, and H. Janicke, “Explainable Artificial Intelligence for Intrusion Detection Systems,” IEEE Access, vol. 10, pp. 69458–69471, 2022.

[14] H. Janicke, L. Maglaras, and M. Ferrag, “Hybrid Explainable IDS for Robust Network Defense,” Computers & Security, vol. 130, 2024.

[15] N. Moustafa and J. Slay, “UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems,” Proc. MilCIS, 2015.

[16] A. Thakkar, A. Soni, and R. Rathod, “A Review of the Advancement in Intrusion Detection Datasets,” Computers & Security, vol. 100, 2020.

[17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion

Detection Dataset and Intrusion Traffic Characterization," Proc. ICISSP, 2018.

[18] Canadian Institute for Cybersecurity, CICIDS2017 Dataset Description, University of New Brunswick, 2017.

[19] University of Surrey Open Research, Machine Learning-Based Intrusion Detection Study Repository, 2023.

[20] ACM Digital Library, Hierarchical State Machine Architectures for Network IDS, 2020.

[21] A. M. Ferrag, L. Maglaras, et al., "Deep Learning for Cyber Security Intrusion Detection Systems – Comparative Analysis," Elsevier Computers & Security, 2020.

[22] G. Apruzzese, M. Colajanni, "Adversarial Machine Learning in Network Intrusion Detection: A Survey," IEEE Communications Surveys & Tutorials, vol. 25, 2023.

[23] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macias-Fernandez, and E. Vazquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges," Computers & Security, vol. 28, pp. 18–28, 2009.

[24] KDD Cup 1999 Dataset, UCI Machine Learning Repository, University of California Irvine, 1999.

[25] NSL-KDD Dataset, University of New Brunswick CIC Repository, 2009.

[26] A. Thakkar, "Advancements in Hybrid Intrusion Detection Models Using Deep Learning," IEEE Access, vol. 11, pp. 101200–101215, 2023.

[27] J. Zhou, X. Li, and S. Zhang, "Edge-Based Lightweight Intrusion Detection for IoT Systems," IEEE Internet of Things Journal, vol. 10, no. 4, pp. 3561–3573, 2023.