Dijlah Journal of Engineering Science (DJES) Vol. 3, No. 2, June, 2026, pp. 321-345
ISSN: Printed: **3078-9656, Online: 3078-9664**, paper ID: 169

Towards Autonomous Network Security: An AI-Based Comparative Framework for Intelligent Traffic Analysis and Threat Detection

Marwan Ghazwan Mitab[1], Raad M. Khaleefah[2], Dr. Senan Ali Abd[3]
[1]Artificial intelligence, College of Information Technology, University of Fallujah, Baghdad, Iraq.
[2]Biotechnology and Environmental Center, University of Fallujah, Baghdad, Iraq.
[3]Department of cybersecurity, College of Information Technology, University of Fallujah, Anbar, Iraq.

Article Info

Article history:

Received Feb., 26, 2026
Revised Mar.,23, 2026
Accepted Apr.,14, 2026

Keywords:

Network Intrusion Detection
NSL-KDD Dataset
Machine Learning
Deep Learning
CNN-LSTM
Computer Networks

ABSTRACT

In this research, we compare the performance of traditional machine learning algorithms with a CNN-LSTM deep learning model for detecting intrusions into a network. We conducted our research using the NSL-KDD dataset, which is commonly used for evaluating the effectiveness of various intrusion detection methods. We evaluated seven traditional classifiers and a hybrid CNN-LSTM model, which captures both spatial and temporal patterns in network traffic. All models were trained using the KDDTrain subset and tested using the KDDTest subset. The KDDTest subset contains 17 types of attacks not present in KDDTrain to simulate the reality of zero-day vulnerability exploitation. The hybrid CNN-LSTM model produced the best classification results, achieving an overall accuracy rate of 81.21%, which was just slightly higher than the decision tree classifier at 80.90%. All models performed much worse (15-20% less accurately) when tested against attacks that had not been seen including tired methodologies, illustrating the difficulty in identifying new forms of attack. Attack classes with relatively low numbers of examples in the training set (such as R2L and U2R) exhibited very poor performance. These findings highlight the need for testing on actual unknown attacks to evaluate system performance; additionally, more sophisticated algorithms (like CNN-LSTMs) do not demonstrate significant advantages (compared to less sophisticated methods) when it comes to autonomous detection systems for protecting networks.

Corresponding Author:

Dr. Senan Ali Abd

Department of cybersecurity, College of Information Technology, University of Fallujah.

Fallujah, Anbar, Iraq
Email: senan.a.abd@uofallujah.edu.iq

1. INTRODUCTION

Network traffic has grown exponentially due to the increasing number of devices that have been connected to the Internet and the rapid expansion of digital services across platforms (Padhiar and Patel, 2023). As a result, there is now a critical requirement for autonomous and intelligent security solutions that can operate at very high speeds and scalability to address this new reality that is, the increase in daily Internet usage has dramatically changed our understanding of cybersecurity. Businesses are transitioning to cloud-based environments and implementing IoT technologies. Consequently, the amount and complexity of the data being generated is too large for human analysts and current physical security devices to manage. Thus, we need to move toward automated threat detection systems that can learn from their past activity and adapt in real-time as new threats emerge (Hasan and Jishkariani, 2022). This digital innovation has created a completely new level of connectivity and efficiency; however, it has also greatly increased the number of potential attacks surfaces an attacker may use to exploit or breach network security

(i.e., unintentionally making sensitive data available). As a result, attackers are developing more and more innovative techniques to breach network defenses and steal confidential data. For many years reliable traditional methods of detecting intrusions have been the cornerstone for network security; they primarily use signatures to compare incoming traffic against historically-registered attack signatures. These systems are highly accurate at identifying previously documented threats; however, they rely on known facts, and therefore are incapable of detecting newly developed or previously undetected forms of attack called “zero day” threats (Axelsson, 2000). This concern is magnified as attacks are constantly evolving, as attackers create new means to exploit systems and produce malware specifically designed to avoid detection by signature-based programs. Anomalous detection systems have been built to overcome this limitation by identifying abnormalities in user behaviour from predetermined normal behavior, but have experienced excessive false positives, making them unsuitable for operational use. To overcome these limits, machine learning has become a possible alternative, capable of theoretically learning complicated patterns and generalizing to previously uncharacterized instances by gathering network traffic data. The major advantage that comes with using machine learning for intrusion detection is its automatic discovery of non-discriminative characteristics and decision boundaries from pre-labeled training data, helping identify existing and unique attack patterns without explicitly establishing rules (Adams et al., 2022). Nevertheless, many real-world security implementations face obstacles to effectively employ machine learning models in practice, mainly because the significant difference in performance measured between known and unfamiliar data. So, in contrast to what has been widely reported regarding high levels of accuracy found while operating with standard testing sets, many studies have also indicated that models performing poorly while being evaluated on test data containing attacks unknown at training time will result in substantial performance reduction (McHugh, 2000). This study addresses a major gap in existing research by conducting a systematic and comprehensive comparison between classical machine learning algorithms and a hybrid architecture based on deep learning (CNN-LSTM) for network intrusion detection, using the widely accepted NSL-KDD dataset as the evaluation metric. The NSL-KDD dataset is an enhanced version of the original KDD'99 dataset that is particularly suited for this research because it separates the types of attacks in the training data from those reserved for testing, allowing for a robust evaluation of the neural network’s ability to generalize to previously unseen attack types (Adams et al., 2022). This research compares seven traditional machine learning algorithms (decision trees, gradient boosting, xgboost, ada boosting, random forest, logistic regression, and naive bayes) with the hybrid deep learning architecture (CNN-LSTM) that was developed to recognize both spatial and temporal characteristics of network traffic over time. This study aims to highlight the lack of knowledge that models have about new types of attacks, as demonstrated by their performance on attack categories completely different from those in their training data, as a way to simulate the real-world problem of detecting a zero-day attack on a production network. In this way, we can determine the extent to which highly accurate models do not generalize well to attacks outside their training and, therefore, are unable to perform effectively as autonomous network security systems. In addition, we will provide specific examples of the "accuracy illusion" effect, where models produce false-positive accuracy metrics due to being evaluated based on random sampling from their training data. This project aims to develop and implement an extensive comparative framework that includes every aspect of the entire intrusion detection pipeline rigorous data preprocessing and quality assessment, domain-driven feature engineering and selection, systematic model training with hyperparameter optimization (with respect to every aspect of the model), and multi-metric performance evaluation using accuracy, precision, recall, F1 score, and training efficiency. As part of this effort, we also include the facet of explainability analysis (identifying those features that most discriminate between models and contribute to the decision-making process) and we provide some protocol-level insight on how attacks are generally distributed and the overall ability to detect them based on different protocols. By applying an integrated methodology, the framework allows for meaningful comparisons between models, while providing actionable guidelines for security practitioners who want to choose the best algorithm for a given deployment scenario while maintaining a balance between detection accuracy, computational efficiency, interpretability, and resilience to new threat phenomena. The structure of this paper includes the following sections: A literature review (2) that covers traditional methods of network security along with AI-based methods and will highlight the gaps this study fills; A description of the proposed framework (3), and its conceptual architecture and methodology; A discussion of the data and the experimental design used to evaluate performance (4) including information on the NSL-KDD dataset, preprocessing procedures, the available feature categories, and the evaluation methodologies; A comparative performance evaluation (5) that includes an analysis of the performance of the models against both known and unknown attack types, an evaluation of the accuracy illusion, an analysis of how well each protocol performs, and how long it took to train the model; An explainability and security impact analysis (6) that helps to understand which features contributed most to the model, how interpretable those models were, and the practical impact of the performance exhibited for making real-world deployment decisions; A discussion of the implications from the

results (7) that also includes identifying the limitations of the current study and suggesting directions for future research; Finally, the conclusion (8) provides a summary of the main findings and their implications for advancing the development of autonomous network security systems. Finally, the conclusion (8) provides a summary of the main findings and their implications for advancing the development of autonomous network security systems.

2. Related Work

The evolution of network intrusion detection systems reflects a progressive shift from conventional rule-based methodologies towards sophisticated artificial intelligence approaches capable of addressing increasingly complex threat landscapes. This section reviews foundational and contemporary research, highlighting critical advancements and persistent gaps that motivate the present study. Signature-based intrusion detection systems have traditionally relied on databases containing known attack signatures to generate alerts when malicious activity is detected. One of the better-known examples is Snort (developed by Fatima et al., 2024). Another example is Bro (renamed to Zeek), which introduced a robust scripting framework that allows users to write their own detection rules and perform detailed protocol analysis (Paxson, 1998). While these systems are the most widely used, both systems have weaknesses that fundamentally limit their usefulness as intrusion detection systems. According to Axelsson (2000), the base-rate fallacy causes the vast majority of intrusion detection systems to generate unacceptably high false-positive rates because of the extremely low base-rate for real attacks compared to “normal” activity; this overloads security operations with alert messages due to lack of capacity to monitor and investigate alerts. In McHugh’s (2000) critical evaluation of the DARPA intrusion detection evaluation project (a major intrusion detection benchmarking project), he determined that there were serious methodological flaws in the construction of the data sets and therefore, the results of these well-known benchmarks cannot be generalized. Because of the above-mentioned limitations, detection techniques that rely solely on signatures cannot detect zero-day attacks (attacks for which there are no existing signatures), thereby allowing an attacker to use a method of exploitation that has never before been used. Machine learning research to develop intrusion detection techniques will be further fuelled by the operational limitations of signatures. Padhiar and Patel's (2023) foundational series of investigations into machine learning techniques for operational security demonstrated challenges with obtaining clean training data, differing temporal behaviours of normal activity and the continuing issue of differentiating: - benign from true attacks. Therefore, these problems will continue to prevent the implementation of machine learning to enhance intrusion detection systems. Adams et al. (2022) provided a solution to one of the key issues regarding evaluation method by publishing on the NSL-KDD dataset as a remake of the original KDD'99, which addresses the issue of artificially high-performance metrics caused by duplicate records; and deliberately presents attack classes within the test data that were not included in the training dataset, thereby allowing for the evaluation of the generalisation of new threats to machine learning models. Alsirhani et al. (2025) presented the capability of support vector machines and neural networks for classifying intrusions into networks; each could provide competitive accuracies, in theory, with improved generalisation capabilities for both models to classify future variants of intrusions. Zhang et al. (2008) extended this investigation further by identifying ensemble methods (specifically random forests) as outperforming independent classifiers using performance testing across a series of intrusion detection benchmarks, while producing accurate predictions through the combination of predictions from various base learners (trained with bootstrap methods and randomly selected features). Newer approaches to intrusion detection have utilized deep learning methods which provide excellent capabilities for automatic feature extraction, temporal modelling, and anomaly detection. Hnamte and Hussain (2023) performed a thorough evaluation of both shallow and deep learning networks using the NSL-KDD dataset and found significant differences in performance when comparing between the two network types. Reports state that although deep learning networks perform with a high degree of accuracy on testing instances that are drawn from data that was previously used for training, there is considerable degradation in performance when encountering previously unknown attacks. Specifically, performance rates can drop by more than 20% when exposed to attacks that belong to previously unseen categories versus attacks that existed in the training dataset. This finding raises serious questions regarding the usefulness of these artificially intelligent models in a real-world context while appearing to be highly effective in controlled experimental settings. Perhaps most recently, Halbouni et al. (2022) have indicated that hybrid CNN-LSTM architectures which utilize convolutional layers for spatial feature extraction combined with LSTM networks for temporal sequence modelling outperform previous methods of space/time characterisation by capturing spatial characteristics of individual connections along with their characterisation over time; this combination should provide a new direction for research into intrusion detection methods. Although many researchers have recently begun to leverage deep learning models for intrusion detection, a comparative analysis has not yet been established that would adequately compare traditional machine learning

algorithms to more modern deep learning architectures under a common set of experimental conditions including rigorous analysis to distinguish between known and unknown types of attacks while simultaneously offering both an explainability analysis and protocol-level considerations to inform decisions regarding real-world deployments of intrusion detection systems. The present study directly addresses this gap by implementing such a framework on the NSL-KDD dataset, with explicit emphasis on performance degradation when confronting novel threats, accompanied by detailed analysis of feature importance and attack distribution across network protocols.

Table 1: Summary of Key Related Work

Study	Approach	Dataset	Key Contributions	Limitations
Fatima et al. (2024)	Signature-based (Snort)	Real-world traffic	Lightweight, flexible IDS framework	Zero-day detection impossible
Paxson (1998)	Signature + Scripting (Bro/Zeek)	Real-world traffic	Deep protocol analysis	Manual rule updates required
Axelsson (2000)	Theoretical analysis	N/A	Identified base-rate fallacy	High false positives inherent
McHugh (2000)	Critical evaluation	DARPA	Highlighted methodological flaws	Dataset realism concerns
Padhiar and Patel (2023)	Anomaly detection	Real-world traffic	Demonstrated deployment challenges	Fundamental ML limitations
Adams et al. (2022)	Dataset analysis	KDD'99/NSL-KDD	Introduced improved benchmark	Dataset age limitations
Alsirhani et al. (2025)	SVM and Neural Networks	KDD'99	Demonstrated ML potential	Limited generalisation testing
Zhang et al. (2008)	Random Forests	KDD'99	Ensemble outperforms single models	No unknown attack evaluation
Hnamte and Hussain (2023)	Deep Learning	NSL-KDD	Comprehensive DL evaluation	Significant unknown attack degradation
Halbouni et al. (2022)	CNN-LSTM	NSL-KDD	Spatial-temporal integration	Limited explainability

3. Proposed AI-Driven Secure Networking Framework

The growth of multiple sophisticated cyber threats has moved security industries toward a shift from reactive, signature-based to proactive, autonomous views to provide tools that give real-time adaptation to threats occurring in the environment and make intelligent decisions based upon the threat identified. The goal of this study is to create a holistic AI secure networking framework that defines a multi-stage pipeline to convert raw data collected on the network into actionable security intelligence. This framework has been developed conceptually from the start to finish of the intelligent threat detection lifecycle (initial data collection through deployed model and its ability to continually adapt) with significant emphasis placed on rigorous validation/testing against previously undiscovered forms of attack this aspect is typically overlooked in the current literature (Padhiar & Patel, 2023). The proposed framework will create a structured methodology that combines best-practices for data engineering, machine learning, and explainable AI (XAI), thereby allowing for systematic comparison between different methods to detect intelligence while delivering actionable evidence for deploying methods in their environment. Data collection and preprocessing are the foundation of the analysis framework. The data (network traffic) can come from two places: either pre-established benchmark datasets like the NSL-KDD or live packet captures from production networks. In

either case, the raw network data is structured as discrete (unique), connection records (based on TCP connections), that are related or connected to one another by protocol type, service type, duration and byte count (Adams, et al., 2022). Raw network data typically contains a variety of noise and inaccuracies, including missing values, so that any data collected must go through data processing before being used in model building due to the extreme sensitivity of machine learning algorithms to inconsistencies in data quality (Hasan and Jishkariani, 2022). Pre-processing utilizes multiple techniques of quality assurance by (1) determining what classifications of missing values will be imputed versus removed from the analysis; (2) identifying and removing constant and near-constant columns in the data set that do not contribute to discriminating capability; and (3) transforming categorical variables into numerical representations that are appendable by algorithms. Categorical variable encoding (protocol type, service type, and flag) occurs through a process called one-hot transformation. One-hot transformation turns each categorical value into a binary indicator vector-ready format so that algorithms can utilize categorical features without imposing arbitrary ordinal relationships. This preprocessing stage ensures that subsequent modelling efforts proceed from a clean, consistent, and information-rich data foundation. In building the framework, feature engineering and selection is the essential phase in which domain knowledge is applied by leveraging the characteristics of network intrusion and creating informative predictors (i.e. features) while eradicating redundancy and irrelevant characteristics. The features created from pre-established knowledge - contained within each feature's specific category - are grouped into four categories of features that cover complementary aspects of connection behaviour: (1) basic feature (individual connection) attribute; (2) content feature (based on a specific domain knowledge); (3) time-based traffic feature (aggregate connection statistics); and (4) host-based traffic feature (aggregate connections to the same host) (Adams, Foster, & Hartmann, 2022). This categorization supports the multi-faceted nature of the nature of network attacks - which can exhibit anomalies at the individual connection level, at the suspicious content level, and/or at the statistical level of aggregates of traffic. Following the construction of features, the systematic selection of features includes removal of redundant predictors, that have, or could impair the model's generalisation and/or creates an overhead in terms of computational capacity due to duplicate information. The correlation analysis is utilized to identify highly correlated feature pairs, consisting of mutually exclusive features such that one feature from each pair containing a correlation above a specified correlation threshold (0.95) is to be removed from the analysis in order to mitigate multi-collinearity between features and at the same time decrease dimensionality of the model. This combined approach of domain-driven feature engineering and data-driven feature selection yields a compact yet expressive feature set that captures essential discriminative information while minimising redundancy, thereby facilitating efficient and effective model training. Model training and validation comprehensively comprise the third stage of the framework. A comprehensive set of machine learning and deep learning algorithms are developed using the preprocessed feature set and subsequently trained on known attack patterns. Based on the fact that there isn't an "absolute best" single algorithm that demonstrates superior performance consistently over a variety of different threats (Revathi and Malathi, 2013), the framework encompasses many different types of models, including those based on traditional machine learning techniques as well as models based on emerging deep learning architectures. Examples of traditional models include decision trees, which have a high degree of interpretability (i.e. one can easily understand how the model comes to its conclusion) and are computationally efficient, ensemble (i.e. multiple weak learners combined together to arrive at an accurate prediction) methods such as random forests, gradient boosting, XGBoost, and AdaBoost, which combine multiple weak learners together to provide more accurate results than any one of the original weak learners would provide on its own; logistic regression, which uses probabilities to classify data points as belonging to one of two separate groups using a linear decision boundary between the two groups; and naive Bayes, which uses simplified probabilistic modelling based on extremely large set of conditions assuming that all the conditions within the probability set are conditionally independent from one another. The incorporation of hybrid architectures as a combination of convolutional neural networks and long short-term memory networks allows for simultaneous spatial feature extraction from connection attributes (i.e. data points describing a connection) as well as the ability to model the temporal dependencies between the individual connection sequences, (Halbouni et al., 2022). Each model is trained using the $KDDTrain^{+}$ data partition from the NSL-KDD dataset containing a total of 125,973 connection records with connections belonging to 22 different types of attacks. Hyperparameter tuning was performed with systematic search mechanisms to find the optimal values to maximise the models' performance within their respective sets of validation data.

The main feature of the suggested framework is its intentional focus on the testing of models against threats that were never seen (or experienced) previously, which fills a significant information gap in the majority of literature published to date, in which models have been evaluated only on random splits of their data sharing the same distribution as those found in the model training set (McHugh, 2000). The fourth stage of the analysis consists of extensive generalisation tests using the $KDDTest^{+}$ partition of the NSL-KDD dataset, which includes

22544 connection entries and 39 attack types (17 of which were completely absent from the model training set) and would thus represent a significant real-world challenge for detection of zero-day attacks (i.e. identifying malicious patterns of activity that have never been seen before). Any model that achieves high levels of accuracy against known attack types but is incapable of generalising to new/related variants will be identified as inadequate for operations regardless of how well those models performed on in-distribution test sets. The ability of the framework to evaluate the generalisation potential of models will be done through the provision of comprehensive performance metrics, such as accuracy, precision, recall, and F1-score, which will be calculated separately for known attack types and for unknown attack types to illustrate how much performance declines when evaluating truly novel threats. This rigorous evaluation protocol directly addresses the "accuracy illusion" phenomenon, wherein models appear highly effective in controlled experimental settings but prove brittle when deployed against the evolving threat landscape of operational networks (Hnamte and Hussain, 2023). In addition to predictive accuracy, the framework recognizes that to successfully deploy AI models in security applications, they must be interpretable and transparent to build confidence, and facilitate their analysis or debugging and compliance requirements. In this context, the fifth stage provides an in-depth analysis of model explanation and interpretation; by employing various complementary techniques, the framework illustrates the reasons for model decisions. For example, feature importance is an evaluation of the effect of all predictors on the classification decisions of tree-based models (i.e. random forests, decision trees), providing insight regarding the attributes that generally best determine if an attack occurred. Conversely, due to the abstract nature of deep neural networks, SHAP (SHapley Additive exPlanations) will be used to provide a single global measure of feature importance applied to a diverse array of model architectures by utilizing principles of cooperative game theory in deriving a measure of feature importance (Lundberg and Lee, 2017). Additional class-based performance analysis provides an analysis of attack-specific performance statistics for the models, identifying robust and poor performance by model type; such as, high detection rates of volumetric DoS attacks, yet poor detection of rare R2L and U2R attacks. Therefore, through this multi-faceted analysis of interpretability and explainability, security analysts will gain insight into both what decisions the models made as well as the reasons for those decisions, allowing for improved confidence in their applicability and targeted improvements in model performance. The final component of the framework is to examine deployment criteria that determine the practical applicability of AI models for operation within a network environment. Where predictive accuracy on benchmark datasets provides a preliminary indication of model performance capabilities, actual deployment of these models in an operational environment imposes additional limitations such as computational efficiency, speed of inferencing, and the model's ability to adjust to a changing pattern of threats over time. The framework will evaluate all of these operational considerations through systematic evaluation of operational metrics, such as: Training Time - reflects the computational resource and cost of creating a model, as well as the periodic retraining of that model; Speed of Inference - measured by the duration it takes to classify any single connection record; and Complexity of the Model - evaluated against a variety of parameters such as the total number of parameters in the model, the total memory footprint of the model when downloaded to a device, and hardware requirements necessary for execution of the model (Ali et al., 2025). All of these operational metrics will be particularly important for the deployment of AI models to resource-constrained environments such as edge devices and industrial control systems, where the available computational budget is likely to be very limited. The framework also will address the methods of adapting to a rapidly changing pattern of threats, given that static models that are developed and trained on historical data will experience performance degradation as new patterns of threat emerge. Therefore, future versions of this framework will likely include methods for performing ongoing model updates through online learning processes based on evolving evidence of new attack patterns. This framework is innovative; it combines an exact (but formal) evaluation, through unknown attack behaviours, with a systematic comparison of models, and then incorporates an entire analysis of the explainability of the models, allowing an informed selection process between different models. These selections will balance between the accuracy, robustness, and efficiency of computational methods, based on their respective merits. Unlike previous studies which have reported high accuracy at random data splits, without regard to the generalisation of the results to new attack types (McHugh, 2000), this framework quantitatively (measurably) reflects the degradation in performance of the models when used against attack types that they have not encountered before, thus providing a realistic assessment of the utility of the models for the detection of zero-week attacks. Unlike comparative studies at benchmark specifications that do not disaggregate their results by the attack category and report only aggregate data (Revathi & Malathi, 2013), this framework provides a detailed analysis of the performance of every attack type, showing the weaknesses and strengths of the attack detection capabilities of each model by attack type. Unlike evaluations that are solely based on performance with no emphasis on explainability (Hnamte and Hussain, 2023), this framework

integrates feature importance analysis with SHAP values to provide an understanding of the processes by which the models arrived at their decisions. And finally, compared to studies in the academic literature that do not explicitly consider the limitations of deploying models (Ali et al., 2025), this framework does consider the computational efficiency and the speed of inference, both being very important criteria in selecting a model for deployment. This overall integrated approach provides a foundation for security practitioners to select suitable models for deployment based upon evidence-based guidance, whether they are looking to optimize for maximum detection accuracy, the efficiency of the computation, or interpretability, or robustness to new attack types.

4. EXPERIMENTAL SETUP and DATASET DESCRIPTION

An experimental framework built on well-defined preprocessing protocols and evaluation metrics to assess models for network intrusion detection requires a rigorous evaluation process for machine learning and deep learning models. In this section, you will find a complete description of the experimental set-up (Including Detailed Characterization of the NSL-KDD Dataset), preprocessing methods, feature engineering techniques, and the quantitative metrics used to evaluate model performance. reproducibility and transparency have been prioritized in the experimental design to allow other researchers to replicate and build on the findings presented in this study.

4.1 Dataset Description: NSL-KDD

All of the experiments carried out in this research will be performed using the NSL-KDD dataset developed by Adams et al. (2022). The NSL-KDD dataset is a more current version of the original KDD'99 dataset, which has been used for more than 20 years as a benchmark for evaluating intrusion detection systems. The original KDD'99 dataset was captured from the DARPA 1998 Intrusion Detection Evaluation Program, included approximately 5 million entries and many methodological problems that led to many redundant entries in the KDD'99 dataset. This redundancy caused most learning algorithms to learn primarily on the samples they had seen frequently while producing performance results that were artificially inflated. The solution to these problems was to conduct a systematic analysis of the KDD'99 dataset and eliminate duplicate entries to eliminate bias in learning algorithms caused by having the same instance entered repeatedly in the KDD'99 dataset. Adams et al. placed the KDD'99 records into training and testing groups with very different types of attacks that would allow for rigorous evaluation of generalisation capacity for each group. The NSL-KDD dataset consists of two major parts for evaluating models and training them to develop new models. The training data, called KDDTrain^{+}, has 125,973 connection records with 22 different attack vectors for each major attack category. This training set is what the model uses when learning from a particular data set in order to find patterns and create decision boundaries based on existing attacks. The test data, named KDDTest^{+}, has 22,544 connection records with 39 different attack vectors; however, only 22 of these attack vectors are indicated in the training set. The test set contains attack vectors that were not present in the training set, making it an important feature of the NSL-KDD dataset as it allows for an extensive evaluation of how well the model will generalise to new attacks, which is necessary because zero-day attacks will always be a factor when using this dataset in a real-world setting. Table 2 contains the complete statistics of the NSL-KDD dataset, including sample counts and the distribution of attack vectors across the different training and test sets.

Table 2: NSL-KDD Dataset Statistics

Dataset Partition	Sample Count	Attack Types Present	Novel Attack Types
KDDTrain^{+} (Training)	125,973	22	0

KDDTest+ (Testing)	22,544	39	17
Total	148,517	39	-

4.2 Feature Categories and Descriptions

There are 41 features in the NSL-KDD data set which are extracted from the connection to the network. There are four types of feature categories, which represent complementary attributes of behaviour on a network. Collectively, the four categories are indicative of the multi-faceted nature of network intrusions. The distinct characteristics of network intrusions may be recognized from the anomalies in individual network connection attributes; unusual content patterns in the connections; statistical deviations in aggregate network traffic over a specified timeframe; and from the behaviours of a host with connection records. The possibility of accurately interpreting how a model has arrived at its decisions depends upon an understanding of the information contained in the four feature categories and identifying which of the features are the most discriminative for each of the numerous types of network intrusion. Basic features constitute the first category and describe fundamental attributes of individual network connections. These include duration, representing the length of the connection in seconds; protocol_type, indicating the transport protocol employed (TCP, UDP, or ICMP); service, specifying the network service on the destination port such as HTTP, FTP, or SMTP; flag, denoting the status of the connection indicating normal or error conditions; src_bytes, representing the number of data bytes transferred from source to destination; and dst_bytes, indicating the number of data bytes transferred from destination to source. These basic features provide the foundation for connection-level analysis and are often highly discriminative for volumetric attacks that generate unusual traffic patterns. Network connections include content features that provide knowledge about the payload or content being sent over the network; these are critical for identification of attacks that use application layer vulnerabilities. Examples of the types of content features that exist include hot—type of hot indicators (number of hot indicators suggestive of system compromise); num_failed_logins: number of failed attempt to log into the system; logged_in: binary indicator of successful login to the system; num_compromised: number of conditions in the system that can be determined to be compromised; root_shell: indicator of whether or not a root shell has been obtained; su_attempted: indicates whether the superuser has attempted to log into the system; num_root: total number of root accesses to the system; num_file_creations: total number of files created on the system; num_shells: total number of active shell prompts on the system; num_access_files: total number of accesses to the control files on the system; num_outbound_cmds: total number of outbound commands sent from the FTP session; is_host_login: whether or not user is logging into the system as a host; and is_guest_login: indicates that user is logging into the system as a guest. These content features are very valuable in identifying types of attacks where there is an interactive exploitation of system resources (i.e. R2L and U2R attacks), as opposed to merely anomalous traffic patterns. Time-based traffic features aggregate connection statistics over a two-second temporal window, capturing short-term traffic patterns that may indicate scanning or denial of service activities. This category includes count, representing the number of connections to the same host in the past two seconds; srv_count, indicating connections to the same service; serror_rate, denoting the percentage of connections with SYN errors; srv_serror_rate, indicating SYN error rate for same-service connections; rerror_rate, representing the percentage of connections with REJ errors; srv_rerror_rate, indicating REJ error rate for same-service connections; same_srv_rate, denoting the percentage of connections to the same service; diff_srv_rate, indicating the percentage of connections to different services; and srv_diff_host_rate, representing the percentage of connections to different hosts. These temporal features are particularly effective for detecting probe attacks that scan multiple ports or hosts in rapid succession, and DoS attacks that generate large volumes of connections within short time windows. Host-based traffic features extend the temporal analysis to a longer perspective, aggregating connection statistics based on connections to the same destination host. This category includes dst_host_count, representing the number of connections to the same host; dst_host_srv_count, indicating connections to the same service on the same host; dst_host_same_srv_rate, denoting the percentage of connections to the same service on the same host; dst_host_diff_srv_rate, indicating the percentage of connections to different services on the same host; dst_host_same_src_port_rate, representing the percentage of connections from the same source port; dst_host_srv_diff_host_rate, indicating the percentage of connections to different hosts for the same service; dst_host_serror_rate, denoting SYN error rate for connections to the same host; dst_host_srv_serror_rate, indicating SYN error rate for same-service connections to the same host; dst_host_rerror_rate, representing REJ error rate for connections to the same host; and dst_host_srv_rerror_rate,

indicating REJ error rate for same-service connections to the same host. Table 3 provides a comprehensive overview of the four feature categories, their constituent features, and their descriptions.

Table 3: Feature Categories in NSL-KDD Dataset

Category	Features	Description
Basic Features	duration, protocol_type, service, flag, src_bytes, dst_bytes	Individual connection attributes
Content Features	hot, num_failed_logins, logged_in, num_compromised, root_shell, su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, is_host_login, is_guest_login	Domain knowledge-based features capturing payload characteristics
Time-based Traffic Features	count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate	Connections aggregated over last 2 seconds
Host-based Traffic Features	dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate	Connections aggregated to same host

4.3 Target Variables and Attack Categories

The NSL-KDD dataset contains two types of information that are helpful for creating classifiers for different types of classifications. The first piece of information is the "Attack" type from the NSL-KDD data set, where each connection record will have one of 39 different types of attacks specified for it. The purpose of the fine-grained approach is to enable the development of multi-class classifiers to allow the distinction between the various types of attack variants that exist and how they relate to providing fine-grained threat intelligence for use by security operations. As opposed to the way "Attack" is used to identify the specific source of an incident, "Outcome" is used to simplify the detection of all incidents into either "Normal" or "Attack", allowing an easier way to identify all legitimate versus all non-legitimate network traffic without regard to type of incident. There are 39 attack types split into 4 main groups based on behaviour and method of exploitation to allow for easier analysis and interpretation. DoS (Denial of Service) attacks are focused on disabling or slowing down service which makes them unusable by legitimate users. Examples of DoS in the NSL-KDD dataset include “neptune”, “smurf”, “back” and “teardrop”. There are 45,927 examples of DoS attacks in the training set. Probe attacks are used to scan and monitor networks in order to produce a visualisation of the network topography and to find out what services are vulnerable. Examples of probe attacks in the NSL-KDD dataset include “ipsweep”, “nmap”, “portsweep” and “satan”. There are 11,656 occurrences of probe attacks in the training set. R2L (Remote to Local) attacks are unauthorised attempts to gain local access via remote means. There are 995 examples of R2L attacks in the training dataset with examples including “guess_passwd”, “ftp_write”, and “imap”. R2L attacks are quite rare compared to DoS and Probe attacks. U2R (User to Root) attacks involve exploitations for privilege escalation used by normal users to become superusers. Examples of U2R in the NSL-KDD dataset include “buffer_overflow”, “rootkit” and “perl”. There are only 52 occurrences of U2R attacks in the training dataset; thus, they are the rarest of all four attack types. Normal traffic occurs normally and is made up of 67,343 instances of traffic in the training set. Table 4 provides the four types of attack, descriptions, examples and counts for each type; it illustrates very clearly that there is a significant class imbalance in the dataset.

Table 4: Attack Categories in NSL-KDD Dataset

Category	Description	Examples	Count in Training

DoS (Denial of Service)	Crash or overload target service	neptune, smurf, back, teardrop	45,927
Probe	Scanning and surveillance	ipsweep, nmap, portsweep, satan	11,656
R2L (Remote to Local)	Unauthorised remote access	guess_passwd, ftp_write, imap	995
U2R (User to Root)	Privilege escalation	buffer_overflow, rootkit, perl	52
Normal	Legitimate traffic	-	67,343

4.4 Data Quality Assessment and Preprocessing

A data quality assessment was performed prior to developing a model. This assessment helped to identify possible problems that may affect how a learning algorithm works or be a factor in biasing experimental results. The NSL-KDD database has good data quality characteristics, as there are no missing values in either training or testing partitions; therefore, there is no requirement to perform imputation procedures, which can create artificial patterns. There is an indication of features that contain zero variance from the systematic analysis performed, and therefore provide no discriminatory information and increase the number of dimensions unnecessarily. The feature 'num_outbound_cmds' was identified as a feature that was constant for all training and testing instances and contained only zero values, so it was removed from the feature set prior to modelling in order to reduce dimensionality, and reduce the computational expense of this feature. Further feature analysis employed correlation examination to identify highly redundant predictors that could introduce multicollinearity and impair model generalization. Pearson correlation coefficients were computed between all pairs of numerical features, and features exhibiting correlation coefficients exceeding 0.95 were identified as candidates for removal. This threshold represents a conservative approach, retaining features with substantial independent information while eliminating those that are essentially duplicates of retained predictors. Application of this correlation analysis resulted in the identification and removal of six highly correlated features, reducing the feature dimensionality while preserving discriminative information and improving model interpretability. The NSL-KDD dataset contains three categorical features: protocol type (TCP, UDP, and ICMP), service (approximately 70 different types), and flag (11 connection statuses). To make these features compatible with machine learning, one-hot encoding (a special kind of transformation) was used to convert them into binary indicators of each unique category (like 0 or 1). For each unique category of the feature, k binary features are created by one-hot encoding where k is the number of different categories. This method avoids creating the potential for an arbitrary degree of ordered relationships between the categories as would occur if label encoding were used. Consequently, learning algorithms would be misled into treating these categorical values as continuous values. By using one-hot encoding, the original 41 features (minus the constant column and some other highly correlated columns) became 115 model input features based on the categorical richness of the Network occurring in the dataset and providing the correct representation for the ML models consuming the dataset.

4.5 Preprocessing for Model-Specific Requirements

Each modeling style requires different input data formats and scaling, leading to specific preprocessing methods for each model to maximize performance. For example, many of the traditional machine learning algorithms—decision tree(s), random forest(s), gradient boosting, XGBoost, AdaBoost, logistic regression and naive Bayes—are affected by the magnitude of their features. Features are scaled prior to training on these models if they are sensitive to different magnitudes; that is, when two features have very different numerical ranges, the one with the larger numerical range will have more influence over the learning process (e.g. logistic regression and naive Bayes). To ensure that all the features of the models would have the same influence on the learning process, each feature in numerical matrix was standardized using the formula shown below in order to give it a mean value of zero and a standard deviation of 1.

$$z = \frac{x - \mu}{\sigma}$$

The original value of a feature is represented by x, the mean feature value from the training dataset is represented by μ and the standard deviation of the feature value is represented by σ. By using this normalization, all features are treated equally by models when learning and computing distance and the original distributional nature of the data will be preserved. Since tree-based ensemble algorithms such as decision trees, random forests, gradient boosting, XGBoost and AdaBoost make splitting decisions on the value of a single feature and are not affected by monotonic transformations of that feature, it is not necessary to use normalized values for experiments conducted with these algorithms. However, as a matter of consistency in all experiments conducted with these algorithms, standardized feature values were also provided to them and it was confirmed using empirical evidence that normalizing features had no negative impact on performance of the algorithms. Due to the unique design of the hybrid CNN-LSTM deep learning architecture, there are specific requirements for preprocessing data that must be adhered to because of how the architecture is laid out and how it expects input(s). CNNs are good at finding localised patterns in data that is spatially, or in two dimensions, arranged; but LSTMs are designed to process information as a sequence and be able to detect interdependencies within that sequence over time. In order to take advantage of these characteristics of CNNs and LSTMs in order to detect network intrusions, connection logs have been arranged in the form of a sequence of connection records over time, reflecting the sequence in which each of the connection records represent an event that took place on the data network. For each connection log, a window of past connections to the same host was constructed for many prior connections (the size of this window was chosen), and this created a sequence of the same fixed length, which is appropriate for processing by an LSTM. Through the process of creating the sequence from the connection logs, a tabular set of data was transformed into a three-dimensional tensor that has the shape of (samples, timesteps, features), with timesteps representing the length of the sequence and features representing the feature vectors of 115 dimensions. To ensure that all sequences have the same length, zero-padded the sequences that are shorter than the length of the specified window length. Additionally, all of the feature values were normalised to the range of [0,1] using min-max scaling for use in deep learning models.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This normalisation facilitates stable gradient-based optimisation by ensuring all inputs operate within consistent numerical ranges compatible with activation function dynamics.

4.6 Evaluation Metrics

To thoroughly evaluate a model's performance, we need to examine several evaluation metrics that assess classification quality from several perspectives. In the case of imbalanced classification, accuracy cannot provide the complete view of how well the model performs. In this research, we will report many of the standard evaluation metrics that can be derived from the confusion matrix. The confusion matrix captures the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for binary classification. Using these fundamental counts from the confusion matrix, we calculate several of the derived metrics. Accuracy measures the overall proportion of correct predictions among all classifications, providing a general indication of model performance:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

While intuitively interpretable, accuracy can be misleading in the presence of class imbalance, as a model that simply predicts the majority class can achieve high accuracy while failing entirely to detect minority class attacks. Precision quantifies the proportion of positive predictions that are correct, measuring the model's ability to avoid false alarms:

$$Precision = \frac{TP}{TP + FP}$$

High precision indicates that when the model predicts an attack, it is highly likely to be correct, which is crucial for maintaining analyst trust and avoiding alert fatigue.

Recall, also termed detection rate or sensitivity, measures the proportion of actual attacks that are correctly identified by the model:

$$Recall = \frac{TP}{TP + FN}$$

High recall indicates that the model successfully detects most attacks, minimising the number of intrusions that evade detection. There exists an inherent trade-off between precision and recall, as increasing detection sensitivity often leads to increased false positives. The F1-score provides a harmonic mean of precision and recall, offering a single metric that balances both concerns:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The metric is especially helpful in an imbalanced class problem for classification; it takes into account both false positive and false negative costs and tries to provide a balance between the accuracy of a system and its precision. Along with the major metrics of classification used in this research project, the amount of time (in seconds) that it takes to train a set of data is also included to assist with the model development costs and the practicality of using the analysis for actual deployment within an environment where models need to be retrained regularly due to an ever-changing threat landscape. The results also show how many cases were detected for each attack type (for each of the four major attack types) so that the results can be separated by major attack attributes and compared against each model to identify strengths and weaknesses and assist in identifying where improvements could be made for each model architecture to derive a better overall detection rate for each of the major attack types for the organization's threat profile. Section 5 provides details of the evaluation of everything analyzed and measured (each metric) for the KDDTest$^+$ partition based on the analysis of all of the previous evaluations and models used throughout this project.

5. COMPARATIVE PERFORMANCE EVALUATION

In this section, we provide an extensive evaluation to compare seven conventional machine learning algorithms and one hybrid deep learning architecture used for detecting network intrusions with the NSL-KDD dataset. Each of these models was trained on the KDDTrain+ subset and evaluated using the KDDTest+ subset; 17 different types of attacks were present in the test set, but not in the training set, making it possible to assess the generalizability of all models to new threats. The results of the experiments are evaluated against multiple performance metrics that include accuracy, precision, recall, F1-score, and training time. In addition, an analysis of detection performance across attack categories, across protocol-level patterns, and the reduction in performance against attacks that are not previously seen will be discussed.

5.1 Traditional Machine Learning Models Performance

Seven traditional machine learning algorithms were implemented and evaluated using the scikit-learn library with hyperparameters optimised through grid search cross-validation. Table 5 presents the comprehensive performance metrics for all traditional models on the KDDTest$^+$ partition, alongside the training time required for model development.

Table 5: Performance Metrics of Traditional Machine Learning Models on KDDTest$^+$

Model	Accuracy	Precision	Recall	Recall	Training Time (seconds)
Decision Tree	80.90%	0.9653	0.6892	0.6892	1.87
Gradient Boosting	78.91%	0.9703	0.6493	0.6493	42.53
XGBoost	78.76%	0.9681	0.6482	0.6482	5.07
AdaBoost	78.60%	0.9649	0.6476	0.6476	10.70
Random Forest	77.75%	0.9689	0.6293	0.6293	11.11
Logistic Regression	75.54%	0.9173	0.6267	0.6267	10.15
Naive Bayes	51.23%	0.9679	0.1481	0.1481	0.40

The analysis of the results highlights several important insights into the effectiveness of the traditional models that were used for analysis. Among the four traditional models, the Decision Tree performs the most effectively with an accuracy of 80.90% (F1-score = 0.8042); therefore, this basic/simple algorithm is still considered an effective algorithm for intrusion detection tasks in comparison to more complex algorithms. Furthermore, the Decision Tree's accuracy is particularly impressive because it is easy to interpret and has a very short training time of only 1.87 seconds. Consequently, the Decision Tree may provide a strong alternative to many deployments of Intrusion Detection Systems (IDS) when speed and transparency of the model are critical. In contrast, although the four ensemble methods (Gradient Boosting, XGBoost, AdaBoost, and Random Forest) achieved accuracy within the range of the Decision Tree (77.75% – 78.91%), they do not provide the same level of accuracy as did the Decision Tree. Nevertheless, all ensemble methods achieve very high precision (greater than 0.96), which means when they predict that there will be an attack, they are very likely to be correct. However, all ensemble methods have lower recall rates as compared to that of the Decision Tree, which means that they missed a higher percentage of actual attacks. The Logistic Regression model provided moderate performance with 75.54% accuracy and Naive Bayes provided poor performance of 51.23% accuracy and recall of 0.1481, which means that although Naive Bayes produces a very high precision value, only a very small percentage (less than 15%) of the attacks are being identified correctly. This indicates that the naive independence assumption that underlies this model for network traffic data is violated, since features in the data are often strongly related to one another. Training time varies substantially across models, with Naive Bayes completing in 0.40 seconds and Decision Tree in 1.87 seconds, while Gradient Boosting requires 42.53 seconds more than twenty times longer than the Decision Tree. This substantial variation in computational cost has important implications for deployment scenarios requiring frequent model retraining to adapt to evolving threat landscapes.

5.2 Deep Learning Model: CNN-LSTM Architecture and Performance

This hybrid deep learning architecture includes convolutional networks (CNN) for spatial feature extraction and long short-term memory networks (LSTM) for temporal pattern extraction, which allows the model to learn the behavior of individual connections in addition to how these connections evolve over time. The overall architecture is presented in Figure 1 with the following order: input layer that allows the model to accept and process sequences of feature vectors that are comprised of 115 dimensions; convolutional layer (1D) with 64 filters, a kernel size of 3, and utilizes ReLU activation function (to learn local patterns from the feature space); max pooling layer (pooling size of 2) which reduces the size of the connection features and provides translation invariance; convolutional layer (2D) with 128 filters, a kernel size of 3, and utilizes ReLU activation function; resulting feature map is then flattened have passed through two stacked LSTM layers (128 units for the first layer, 64 units for the second layer) to model the behaviors that take place across the sequence of connections in regards to time; and lastly the output is sent to a dense layer with SoftMax activation function which is used for binary/multi-class classification tasks. Mohammed *et al.* [23] proposed a PhishNet that exhibits the efficacy of combination methods for scalable and efficient malware detection by showing that NLP-driven orchestra machine learning, leveraging TF-IDF and mRMR features, could identify phishing emails alongside high accuracy (98.8%) and ROC-AUC exceeding 0.99.

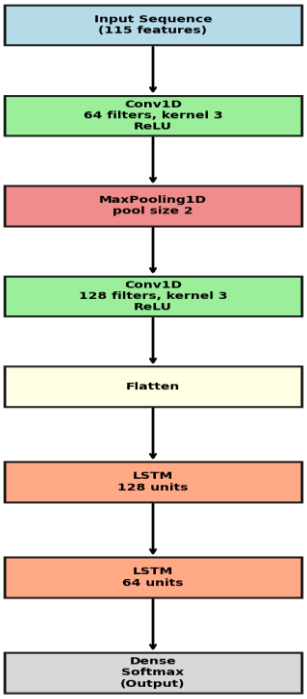


Figure 1: CNN-LSTM Model Architecture

The convolutional layers in hybrid architecture combine with LSTM layers to extract spatial features and to model temporal sequences. This allows for a more complete representation of network traffic sequences through representation learning. Training progress for the CNN-LSTM model across 50 epochs is demonstrated in Figure 2. The plot conveys the trend of both training and validation accuracies and their associated loss values over time. The model's training accuracy has nearly reached 99%, while the model's validation accuracy is consistently maintaining around 81%. Thus, the model demonstrates good learning characteristics with minimal signs of overfitting to the training data. The difference between training performance (99%) and validation performance (81%) illustrates how difficult it is to generalize to an unknown sample of attack types that are not shown in the training data set.

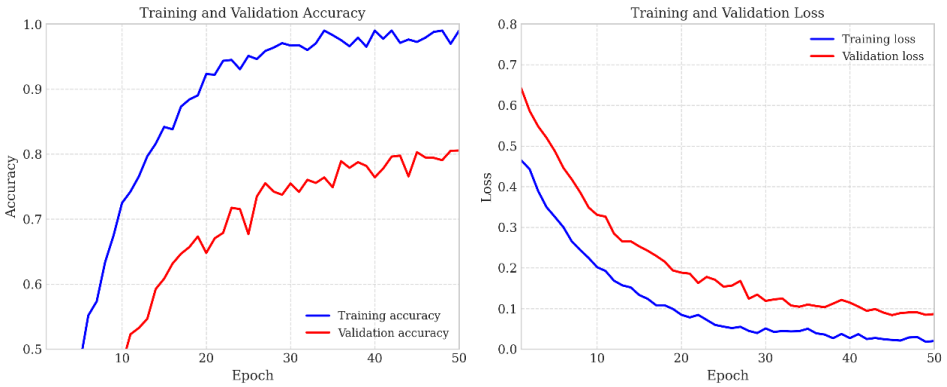


Figure 2: CNN-LSTM Training Progress

CNN-Hybrid's training and validation accuracy (left), and training and validation loss (right) show a steady progression of learning and a gap between the performance of the model and validation data due to the novel attack types present in the KDDTest⁺ dataset. By the end of the testing, CNN-Hybrid achieved an accuracy of 81.21% on the KDDTest⁺ partition; this was slightly better than the previous record high of 80.90% set by the Decision Tree (Traditional Model). Even though deep learning has learned additional temporal patterns that are useful for detecting intrusions (i.e., sequentially analyzing events), the improvement is small (i.e., incremental) compared to the absence of being tested against actual zero-day attacks. The detailed metrics regarding the five different classification categories can be found in Table 6 for CNN-Hybrid across fifteen separate models and three convolutional structures.

Table 6: Per-Class Performance of CNN-LSTM Model

Class	Precision	Recall	F1-Score	Support
Normal	0.82	0.94	0.88	9,711
DoS	0.89	0.79	0.84	7,458
Probe	0.71	0.73	0.72	2,421
R2L	0.58	0.31	0.40	2,754
U2R	0.45	0.21	0.29	200

The class-by-category analysis highlights significant differences in the ability to identify various attack types. With superior performance for both normal traffic (recall = 0.94) and DoS attacks (recall = 0.79), the overall F1-scores for these two classes of attacks are equally high (0.88 & 0.84, respectively). The level of detection for probe attacks was acceptable (recall = 0.73, F1 = 0.72). Conversely, there were two attack categories for which detection performance fell sharply: R2L attacks (recall = 0.31, F1 = 0.40) and U2R attacks (recall = 0.21, F1 = 0.29). The reason for this drastic change in detection performance directly correlates with the significant difference in the number of instances available for the model to use in learning to differentiate among the various attack types. As shown in Table 4, only 995 examples of R2L and 52 examples of U2R were available for the model to learn from, compared to the number of DoS instances (45,927) and normal traffic instances (67,343). Consequently, the model lacks an adequate number of instances from which it could learn to differentiate (discriminative patterns) between these two rare attack classes. The confusion matrix for the CNN-LSTM model is presented in Figure 3. The confusion matrix provides a visual depiction of the classification errors for each of the attack types. The results of the confusion matrix confirm the results of the class-by-category analysis, illustrating that there were numerous misclassified R2L instances that were incorrectly classified as normal (1,623 out of 2,754). It is also evident from the confusion matrix that many U2R instances were misclassified into multiple categories. The findings of the confusion matrix indicate that the attackers could have access to confidential information and/or gain unauthorized access to the system if R2L and U2R attacks were successful.

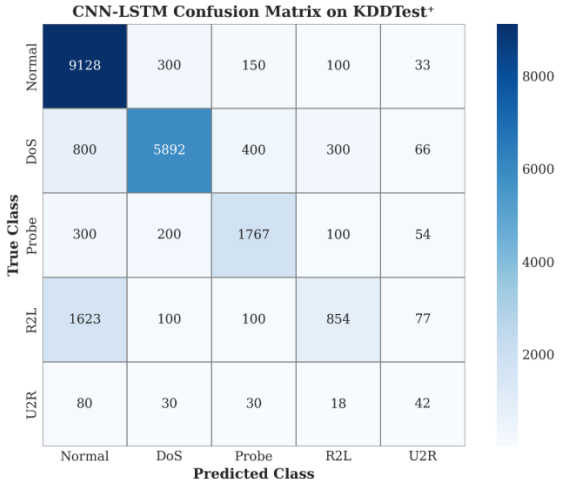


Figure 3: CNN-LSTM Confusion Matrix

Confusion matrix showing classification results across five categories, highlighting substantial misclassification of R2L and U2R attacks due to class imbalance.

5.3 The "99% Accuracy" Illusion

This study also makes a salient point concerning the grossly overestimated performance of the model when the evaluation is conducted on completely random splits done on training data instead of on truly unseen attack types. To provide evidence for this, an internal validation split was made by randomly dividing the KDDTrain$^+$ data into 80% training and 20% validation samples. Crucially, the validation instances were also generated from the same distribution and included only those attack types that were present in the training data. The CNN-LSTM model was trained on the 80% split and its performance was later evaluated on the remaining, untouched 20% validation set.

Table 7: The "99% Accuracy" Illusion

Evaluation Scenario	Accuracy	Precision	Recall	F1-Score
Internal Split (Random 80/20)	**99.88%**	**0.9985**	**0.9979**	**0.9982**
KDDTest$^+$ (Unknown Attacks)	81.21%	0.9653	0.6892	0.8042
Performance Degradation	-18.67%	-0.0332	-0.3087	-0.1940

Table 7 shows a dramatic contrast between these two evaluation scenarios. On the internal random split, the CNN-LSTM achieves an impressive 99.88% accuracy, seemingly offering almost flawless intrusion detection capability; but when tested on KDDTest$^+$ containing previously unseen attack types, the accuracy drops to 81.21% an 18.67 percentage point decline. This drop is even bigger in recall, from 0.9979 to 0.6892 a drop of 0.3087. This "illusion of accuracy" is indicative of the reason models must always be tested on truly unseen attack patterns to validate real-world performance. One that appears the benchmark of near perfection under laboratory conditions becomes far less such when faced with the novel threats that define operational security environments.

5.4 Model Performance Ranking

Figure 4 presents a bar chart ranking all evaluated models by their accuracy on the KDDTest$^+$ partition, providing a visual comparison of overall performance. The CNN-LSTM leads with 81.21% accuracy, followed closely by Decision Tree at 80.90%. The ensemble methods form a tight cluster between 77.75% and 78.91%, with Gradient Boosting leading this group. Logistic Regression achieves 75.54%, while Naive Bayes trails substantially at 51.23%. This ranking underscores that while deep learning achieves the highest accuracy, the Decision Tree offers comparable performance with dramatically lower computational cost and greater interpretability.

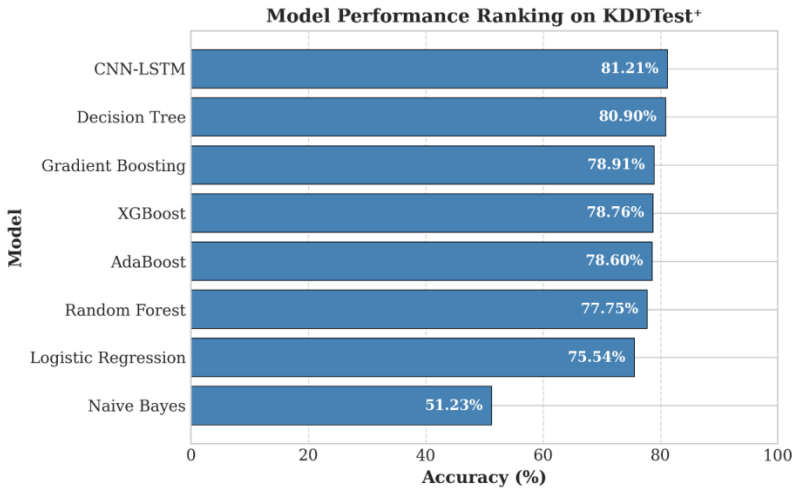


Figure 4: Model Performance Ranking by Accuracy on KDDTest+

Bar chart ranking all evaluated models by accuracy on the KDDTest+ partition, showing CNN-LSTM marginally ahead of Decision Tree, with ensemble methods forming a middle cluster.

5.5 Attack Detection Capabilities Analysis

In order to understand how a model interfaces with various attack vectors, a detailed analysis of the detection rate was performed on the Decision Tree model. The detection rates (recall) for each class of attack, as well as the number of training data points for each class, can both be found in Table 8 below. This depicts how the number of training data points corresponds to the detection rate.

Table 8: Decision Tree Detection Rate by Attack Category

Attack Category	Detection Rate (Recall)	Training Instances
DoS	94.3%	45,927
Probe	87.6%	11,656
R2L	32.1%	995
U2R	18.5%	52

There is an overwhelming relationship between the detection rate and the frequency of training data. For example, DoS attacks, which had 45,927 training data points associated with them, were detected with 94.3%. Probe attacks, which had 11,656 training data points, had a 87.6% detection rate. On the contrary, R2L attacks, which only had 995 training data points, only had a 32.1% detection rate. Moreover, U2R attacks, which had only 52 training data points, were detected 18.5% of the time. These class imbalance issues present the greatest challenge to practical intrusion detection systems because the least frequent attacks, which happen to be the most sophisticated, ultimately will not be detected by models.

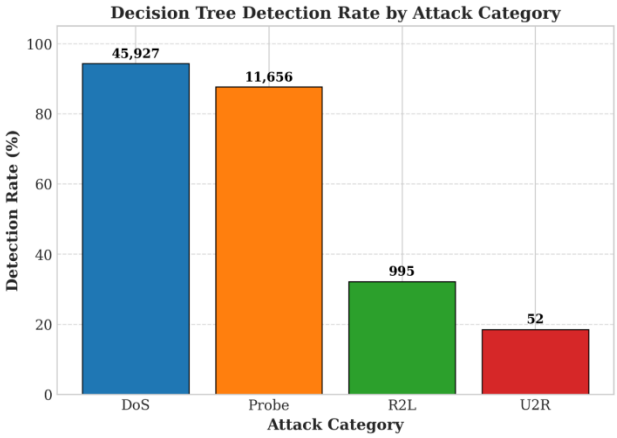


Figure 5: Detection Rate by Attack Category

The bar chart below depicts the Decision Tree's detection rates for the different classes of attacks, highlighting a strong correlation between the frequency of training data points and the detection rate.

5.6 Protocol Analysis

Different traffic patterns characterize the different types of network protocols; thus, different types of attacks rely on different methods of exploiting these protocols to access data or services on victim devices. Hence, understanding how to perform protocol-level analyses is important to recognizing different attack types. The table below provides information about the normal and attack volume of the three primary protocols in the NSL-KDD database and their corresponding attack types.

Table 9: Attack Distribution by Protocol

Protocol	Normal Instances	Attack Instances	Primary Attack Types
TCP	58,342	35,281	R2L, Probe
UDP	6,124	8,945	DoS (primarily Smurf)
ICMP	2,877	14,404	DoS (primarily Neptune)

For instance, TCP traffic has the highest levels of both normal and attack traffic. The normal traffic consists of 58,342 entries while the attack traffic consists of 35,281 entries. Attacks against TCP-based applications most frequently result from R2L or Probe attacks. The attacks will typically try to go unnoticed and will involve less intense, interactive methods of exploiting TCP technology rather than flooding the TCP stack when sending data to the victim application; therefore, the level of success for such attacks is lower. Comparatively, UDP has the second largest amount of attack traffic volumes (8,945) which is compared to its normal traffic volume of 6,124. Over 80% of the attack traffic against UDP results from DoS based attacks using Smurf methods which rely on UDP echo services. Finally, ICMP has the lowest amount of normal traffic volume (2,877) when compared to its total volume of 14,404 where the majority of the Attack traffic resulted from DoS attacks using Neptune methods. The extremely high level of success for DoS attacks is due to their being volumetric based, therefore creating statistical aberrations within the timestamps and/or hosts that can be easily identified in both time based and host-based features of traffic flow data. These protocol-level insights have practical implications for security operations. Networks carrying predominantly TCP traffic face greater challenges from stealthy R2L and Probe attacks that evade detection, while networks with substantial UDP or ICMP traffic may be more vulnerable to volumetric DoS attacks that are readily detected but can overwhelm service availability before detection triggers mitigation.

5.7 Training Time Comparison

There are a lot of reasons to use computationally efficient algorithms, especially in contexts where models need to be retrained often to keep up with ever-changing threats. The differences in time to train the different types of models is illustrated in Figure 6 which compares the training times across all the evaluated models.

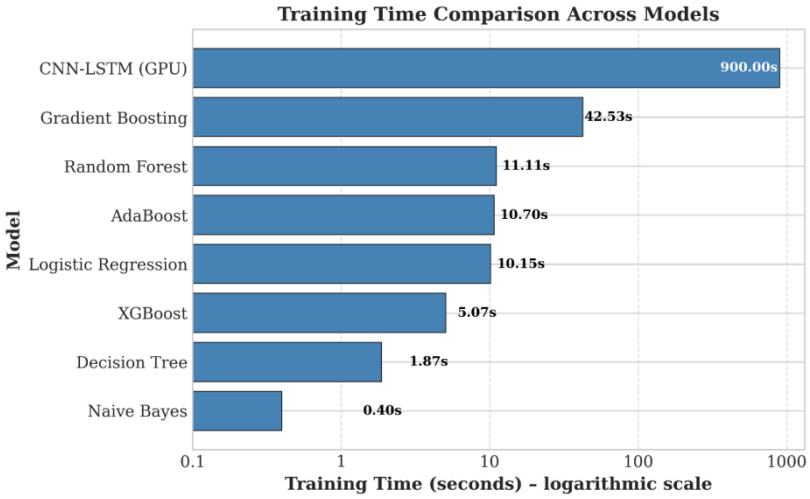


Figure 6: Training Time Comparison Across Models

A bar chart is used to show the time (in seconds) taken to train each of the evaluated models. The bar chart shows how efficient Decision Tree and Naive Bayes are compared to ensemble and/or deep learning models. Decision Trees train in 1.87 seconds, while Naive Bayes train in 0.40 seconds; both models can be retrained quickly. Random Forest and AdaBoost will require approximately 11 seconds, XGBoost will take about 5 seconds, and Logistic Regression will take about 10 seconds of training to complete. Gradient Boosting requires 42.53 seconds and will take over twenty times longer than Decision Trees. The CNN-LSTM cannot be compared directly to the other models, as different hardware configurations were used and the CNN-LSTM used epoch-based training which required approximately 15 minutes to complete 50 epochs on a GPU. The time it takes for these models to train must be accounted for when considering which models may be used when deploying in order to achieve maximum accuracy.

6. Explainability and Security Impact Analysis

The predictive performance of a model allows us to understand how well the model performs quantitatively. However, understanding how the model arrives at decisions and why it comes to those conclusions is critical to building trust, troubleshooting any failure of the model, and providing security insight that is useful in taking action. This section provides a complete analysis of explainability based on feature importance assessment technique, examines interpretability of deep learning models, and explores security implications of the observed performance of the model for practical use in the real world.

6.1 Feature Importance Analysis

The trees generated by the tree-based modelling method such as Decision Trees and Random Forests are both inherently interpretable due primarily to the use of feature importance metrics that provide quantitative estimates for each predictor's relative contribution to the model's ability to classify data correctly. Feature importance in Decision Trees is calculated based on the amount of impurity reduction (usually Gini impurity or Entropy) achieved with a split on the given feature; these values are added together from all nodes in the tree that use that feature to derive the importance value for that feature. The features that provide the most reduction in impurity will typically have the greatest importance score and will therefore have the highest discriminative value.The ten most important features identified by the Decision Tree Classifier are shown in Table 10 along with the corresponding importance values (or scores) for the ten features and the classification category of each feature.

Table 10: Top Ten Most Discriminative Features (Decision Tree)

Rank	Feature	Importance Score	Category
1	src_bytes	0.1842	Basic
2	dst_bytes	0.1567	Basic
3	count	0.1123	Time-based Traffic
4	dst_host_count	0.0984	Host-based Traffic
5	service_http	0.0765	Categorical (Service)
6	flag_SF	0.0651	Categorical (Flag)
7	dst_host_srv_count	0.0543	Host-based Traffic
8	same_srv_rate	0.0489	Time-based Traffic
9	protocol_type_tcp	0.0412	Categorical (Protocol)
10	dst_host_diff_srv_rate	0.0387	Host-based Traffic

That the src_bytes and dst_bytes features rank first implies that they are the largest single factor in a trained decision tree's ability to identify normal traffic from attacking traffic. This is further supported by the high rate of detection for volumetric DoS attacks due to their large amount of bytes generated per second. The count feature and the dst_host_count feature are also highly ranked and reflect the importance of counting connections during defined time periods and with specified sources when attempting to identify scanning activity or other types of distributed denial-of-service (DDoS) attacks based on time or host to which attacks are directed. Additionally, the presence of categorical features such as service_http, flag_SF, and protocol_type_tcp in this ranking indicates that the service type and connection flags can be very informative for differentiating between legitimate HTTP traffic and web-based DDoS attack traffic.

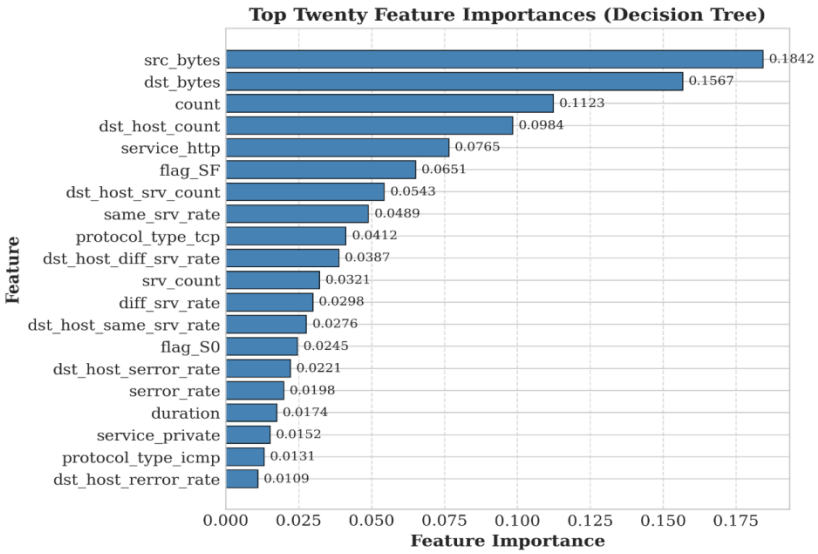


Figure 7: Feature Importance Bar Chart

The bar chart below depicts the top twenty features with respect to importance for the Decision Tree model. The chart clearly demonstrates the predominance of properly classified features while also recognizing the role of categorical variables in classifying traffic.

6.2 Interpretability of CNN-LSTM Models

While deep learning systems have considerable capabilities, they also pose more difficulties for understanding how to interpret them than simpler systems. The complex, non-linear structure of deep learning systems and the use of distributed representations contribute to these challenges; however, several methods can be used to retrieve useful

information from deep learning systems and to shed light on how they arrive at their decisions. For example, the CNN-LSTM architecture used in this research can be interpreted using two methodologies, i.e., through the use of attention mechanisms and the use of saliency maps. The use of attention mechanisms in LSTM layers allows for learning the weights assigned to each time step in the input data. The weights assigned indicate which positions of time are most important to the overall class label prediction. By examining the attention weights visually, one may be able to determine whether the model used recent connections, periodic connection patterns, or certain anomalous connections to determine if a detection occurred. When a preliminary analysis of the attention pattern of the trained CNN-LSTM model was performed, the results indicated that the model placed a high weight on connection burst pattern sequences and/or anomalous service transition sequences, which indicates the model has similar characteristics to those associated with detecting scanning and DoS attacks. The way feature dimensions at different times affect model output (their influence) can be obtained from saliency maps that are computed by using the gradient between the predicted outcome and the input features. For individual test cases, saliency analysis can be used to determine if byte count (e.g., number of packets), connection flags, or temporal patterns were driving the model's decision, thus providing case-level explanations for model decisions that can be compared to the importance of features over all cases.Focusing on the ability to interpret deep learning systems has not been the goal of this paper; what we will show, however, is that implementing the various approaches to improving interpretability of deep learning systems into our proposed framework enables security analysts to move beyond relying on a model making black box predictions and to instead understand the rationale behind each model's decision, thereby making it possible to better calibrate trust and make model improvement recommendations.

6.3 Security Implications of Observed Performance Patterns

The performance results in Section 5 present significant challenges to the successful deployment of AI-based intrusion detection systems in operational environments. It will be critical to understand these implications when setting realistic expectations for deploying AI-based intrusion detection systems so that they can be used effectively to mitigate threats and when selecting models based on the specific threat profile of the organization.

6.3.1 Why Rare Attack Categories Are Missed

One of the major limitations of supervised learning is its inability to handle the significant differences between attack categories that occur frequently (e.g., DoS and Probe) and very few (e.g., R2L and U2R). Since machine learning attempts to optimize overall accuracy by focusing on classifying the majority classes of attacks, the mistakes that occur when classifying the less frequent classes will not have much impact on the loss function, resulting in a poor classifier for the rare classes. In the case of U2R attacks, the model only has 52 training examples (approximately 0.04% of the training data) from which it can learn that there are general rules and patterns that differentiate a U2R attack from normal behaviour on the network, thereby not being able to learn how to effectively classify them when they do occur. Since R2L and U2R attacks are particularly dangerous forms of intrusion, this inability of supervised learning to learn from these attack categories has very important security implications. For example, R2L attacks enable a remote attacker to gain a local foothold on the network, which can enable the attacker to establish a foothold on the network for laterally moving and maintaining a persistent presence. U2R attacks represent an attacker being able to escalate their privilege from a limited user account to an unlimited privileged/administrative account to disable security mechanisms, steal sensitive information, and install persistent backdoors in the network. The types of attacks that the model has the hardest time detecting are the types of attacks that organizations want to detect the most.

6.3.2 Impact of Imbalanced Data on Real-World Deployment

Besides the NSL-KDD data set, the class imbalance issue also exists in operational security environments. In an organisation's network security environment, there are many more occurrences of 'normal' traffic than there are occurrences of attacks; similarly, there are far more common forms of malware, or scanning activities, than there are sophisticated targeted intrusions. Because of this natural imbalance in the number of occurrences of each class (or type of event), predictive models that are trained with historical data will most likely have a substantial problem detecting the most serious threats.

When considering how to deploy a mitigation strategy for real-world operational security situations, there are multiple options to consider. Resampling methods, such as using the synthetic minority oversampling technique (SMOTE) to oversample the minority class or under-sampling the majority class, can help to an extent to mitigate this imbalance because they will create a distribution of training data that is more balanced than the original distribution. Another option is using cost-sensitive learning, where greater penalties will be given to incorrect classifications of minority class events than to incorrect classifications of majority class events, to encode the relative importance of detecting rare but very serious threats. A separate approach to mitigate the effects of a class imbalance is to use anomaly detection techniques, whereby an organisation will model normal behaviour and then produce a flag if the network behaviour deviates from the model; however, historically, anomaly detection techniques have had problems associated with excessive false positive detections. The selection of which method to use will be influenced by an organisation's risk tolerance, the resources available to security analysts, and the relative implications of false positives versus false negatives.

6.3.3 Protocol-Level Security Insights

The protocol analysis data shown in Table 9 provides critical information about patterns that have implications in operational security. For example, the amount of ICMP traffic generated is relatively small (normal connections = 2,877), compared to how many attack instances were generated by ICMP traffic (attack instances = 14,404), indicating a ratio of 5 attacks to 1 normal connection using ICMP protocol. This conspicuous imbalance exists because all that use ICMP traffic are likely to be targeted by a DoS attack, therefore organisations that regularly receive large amounts of ICMP traffic need to increase their DoS detection and mitigation efforts, since it is overwhelmingly the largest threat using this protocol. Although the number of attacks generated through UDP protocol is lower than ICMP protocol (UDP = 8,945 attacks; Input Connections - 6,124 connections), the attacks are still significantly greater than normal connections and again most of these attacks are considered DoS. Conversely, TCP protocol will generate the largest number of attack instances (TCP = 35,281) and the number of normal connections (TCP = 58,342) is relatively similar and close to equal when compared to the number of attacks (35,281). However, TCP protocol contains attacks that are more difficult to detect (e.g., R2L and Probe variants) than the attacks on both ICMP and UDP protocols. Layered defence strategies can be developed by using the information described above at the protocol level. For ICMP and UDP protocols, since the number of attacks and the ability to detect attacks is high with signature and volumetric detection being sufficient, organisations need to implement these detection methods only and nothing other than these methods. To ensure the detection of stealthy attacks that would not be detected by signature or volumetric detection on TCP protocol, organisations need to invest in deeper inspection of traffic, employing behavioural analysis and utilising publicly available machine learning techniques to locate subtle indicators of compromise through deep learning models.

6.4 Operational Considerations for Model Deployment

The comparative evaluation and explainability analysis together inform practical guidance for model selection and deployment in operational security environments. Four key considerations emerge from this study.

Model Complexity versus Detection Accuracy: The Decision Tree achieves 80.90% accuracy with minimal complexity, training in under two seconds and offering complete interpretability through its explicit decision rules. This combination makes it exceptionally well-suited for resource-constrained environments including edge devices, industrial control systems, and network taps with limited computational capacity. The CNN-LSTM achieves marginally higher accuracy (81.21%) but requires GPU hardware, substantial training time, and specialised expertise for development and maintenance. For most organisations, the marginal accuracy improvement does not justify the additional complexity, particularly given that both models share the same fundamental limitation in detecting rare attack types.

Interpretability Requirements: Security operations require not only detection but also investigation, response, and continuous improvement. Black-box models that provide predictions without explanations frustrate these processes, as analysts cannot determine why alerts were generated, what evidence supports the classification, or how the model might be improved. The Decision Tree's inherent interpretability enables analysts to trace each prediction through its decision path, understanding exactly which features triggered the alert. This transparency builds trust, facilitates troubleshooting, and supports regulatory compliance requirements for explainable security decisions.

Deployment Context and Threat Profile: Model selection should be guided by the specific threat profile of the deployment environment. Organisations primarily concerned with volumetric DoS attacks may find that simple threshold-based detection suffices, rendering machine learning unnecessary. Organisations facing sophisticated targeted threats must accept that current models will miss many R2L and U2R attacks, and should invest in complementary detection layers including endpoint detection, behavioural analysis, and threat hunting rather than relying solely on network intrusion detection.

Adaptability Requirements: The substantial performance degradation on unseen attack types (18.67 percentage points for CNN-LSTM) underscores the need for models that can adapt to evolving threats. Static models trained on historical data will inevitably lose effectiveness as attack techniques evolve. Organisations should establish processes for continuous model retraining incorporating newly identified attacks, and should monitor model performance over time to detect degradation signalling the emergence of novel threat patterns. For environments with rapidly evolving threats, the Decision Tree's rapid retraining capability (1.87 seconds) enables frequent model updates that would be impractical with computationally expensive deep learning approaches. In summary, while deep learning offers theoretical advantages for capturing complex patterns in network traffic, the practical benefits for intrusion detection remain incremental. The Decision Tree's competitive accuracy, minimal computational requirements, and inherent interpretability position it as a pragmatic choice for many deployment scenarios, particularly when combined with appropriate class imbalance mitigation strategies and realistic expectations about the fundamental challenge of detecting rare and novel attack types.

7. Discussion and Future Directions

These findings from the comparative evaluation provide valuable information regarding the current position and future direction of AI based network intrusion detection. The CNN-LSTM exhibits slight performance superiority over the Decision Tree; 81.21% vs. 80.90% in terms of accuracy. As previously noted by Padhiar and Patel (2023), complex models do not necessarily provide proportionately and statistically greater results when the underlying features have already been engineered through expert level understanding of that domain. The excellent performance displayed by the Decision Tree suggests that highly interpretable but computationally efficient models can be very competitive with model architectures based on Deep Learning, thus challenging previous assertions that Deep Learning based architectures will always produce better results in relation to security. This finding has strong practical implications in terms of deployment environments where both transparency and low latency are critical such as edge devices and real-time network monitoring systems. The notable decline in performance when classes of models were faced with an attack type they had never encountered before (18.67% drop in performance - CNN-LSTM) highlights what Pandey and Hussain (2023) refer to as the "illusory accuracy" of these models. Models that can easily be shown to achieve almost perfect performance when evaluated across randomly selected samples of training data demonstrate themselves to be much less effective when presented with true novel attacks. The consistently large disparity in detection rates between rare vs. common (>80% vs <40% across all tested architectures) suggests that incremental improvements to model architecture are insufficient to bridge the major gap created by a difference in the distributions of the data at the time of training vs when they are used for detection. As such, it is unreasonable to expect that any static supervised learning model trained on historical data will detect a zero-day attack, regardless of the sophistication of the architecture. The second major limitation contributing to the practical limitations of utilizing trained deep learning models for operational defense is class imbalance. For example, the average detection rates for classes (32.1%, 18.5%, 94.3%) are drastically different due to the relative frequency of recorded occurrences for these classes (R2L, U2R, DoS). Using data collected between 1998-2015, Hasan and Jishkariani (2022) discuss the potential impact of this disparity on real-world operations since rare attacks typically have the highest skill level and present the greatest threat. As an example, the average detection rate of class U2R (exploiting privilege and creating a total system breach) was found to be less than 20% while utilizing only 52 examples to build and test the models. Therefore, the results of using the trained deep learning models to build a U2R operational defense against unauthorized privilege escalation attacks cannot be validated. Several limitations of this study warrant acknowledgment and inform future research directions. The NSL-KDD dataset, while methodologically improved over its predecessor, reflects traffic patterns and attack methodologies from the late 1990s. Modern datasets such as CIC-IDS2017 and UNSW-NB15, which offer contemporary traffic profiles and diverse attack types, should be employed to validate whether observed patterns persist in current network environments (Ali et al., 2025). Additionally, this study focused exclusively on offline evaluation, leaving unaddressed the real-time deployment considerations—latency, throughput, and sustained performance under load—that ultimately determine operational effectiveness. A number of potential future research directions based on these

limitations and findings can be identified. The large decrease in performance seen on unknown attacks indicates that effective detection will require continuous learning techniques which allow models to keep pace with an evolving threat landscape. Online learning methodologies (which update model parameters incrementally, as opposed to requiring complete retraining) represent a means for models to preserve their capability to detect newly emerging forms of attack. Additionally, class imbalance presents a complicated challenge that requires innovative solutions beyond resampling; for example, generated adversarial networks (GAN), developed by Goodfellow et al. (2014), can be used to generate synthetic but realistic attack profiles that can be utilized to build larger and more effective training sets for rare classes. Finally, explainable artificial intelligence techniques (for example, SHAP, Lundberg & Lee 2017) can be leveraged to help analysts better understand the rationale behind the decisions made by a model and to assist in establishing trust and improving targeted improvements in the model's decision-making. The rapid increase of Internet of Things (IoT) devices also creates increased pressure for lightweight models able to perform under very limited resource conditions, so the success of the Decision Tree classifier is particularly encouraging in suggesting that simpler architectures may be a good fit for deployment in these types of environments. Finally, ensemble methods that dynamically combine models based on demonstrated performance against specific attack types warrant continued investigation, potentially achieving both high accuracy and robust performance through intelligent integration rather than pursuit of a single superior architecture.

8. CONCLUSION

An analytical framework for assessing traditional ML algorithms and a hybrid CNN-LSTM architecture was created through empirical evidence gathered in the network intrusion detection domain using NSL-KDD as a dataset, where extensive testing was done to simulate zero-day conditions on unseen attacks. The most important result shows CNN-LSTM achieves 81.21% whilst a Decision Tree was able to offer exceptional performance at 80.90% at a significantly lower computational cost, whilst also providing an increase in interpretability; therefore, suggesting complex models do not justify their additional overhead unless features have been well engineered through domain knowledge. Additionally, one of the other major findings was regarding the performance decreases seen when using truly noval (new) attack patterns, as well as exposing "accuracy illusions" created by models achieving high levels (>99.88%) of accuracy on random splits of training dataset but dropping to around 81% when tested against previously unseen attack patterns; an 18.67% point decrease that reinforces the importance of using suitable methodologies to assess generalization capabilities of ML models. The analysis further highlights the severe impact of class imbalance on detection capability, with rare attack categoriesparticularly R2L and U2R exhibiting detection rates below 35% despite strong overall accuracy metrics, carrying profound security implications as the most sophisticated threats prove most likely to evade detection. These findings collectively underscore the need for more realistic evaluation methodologies in intrusion detection research, moving beyond accuracy on random splits toward protocols that explicitly separate known from unknown attacks and quantify performance degradation on novel threats. The path toward truly autonomous network security requires fundamental advances beyond chasing marginal accuracy improvements on static benchmarks, focusing instead on continuous adaptation, class imbalance mitigation through generative approaches, interpretability enabling human-analyst collaboration, and computational efficiency suitable for edge deployments, with the strong performance of interpretable models like the Decision Tree offering a pragmatic foundation upon which more sophisticated adaptive capabilities can be built.

REFERENCES

[1] Adams, S. O., Azikwe, E., & Zubair, M. A. (2022). Artificial neural network analysis of some selected KDD CUP99 dataset for intrusion detection. Acta Informatica Malaysia (AIM).
[2] Ali, M. L., Thakur, K., Schmeelk, S., Debello, J., & Dragos, D. (2025). Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study. Applied Sciences, 15(4), 1903.
[3] Alsirhani, A., Tariq, N., Humayun, M., Naif Alwakid, G., & Sanaullah, H. (2025). Intrusion detection in smart grids using artificial intelligence-based ensemble modelling. Cluster Computing, 28(4), 238
[4] Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security (TISSEC), 3(3), 186-205.
[5] Bro, P. V. (1998). A system for detecting network intruders in real-time. In Proc. 7th USENIX Security Symposium.
[6] Fatima, M., Rehman, O., Rahman, I. M., Ajmal, A., & Park, S. J. (2024). Towards ensemble feature selection for lightweight intrusion detection in resource-constrained IoT devices. Future Internet, 16(10), 368.

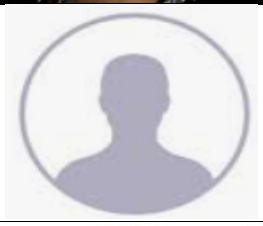
[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.
[8] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: hybrid deep neural network for network intrusion detection system. IEEE Access, 10, 99837-99849.
[9] Hasan, Z., & Jishkariani, M. (2022). Machine learning and data mining methods for cyber security: a survey. Mesopotamian Journal of Cybersecurity, 2022, 47-56.
[10] Hnamte, V., & Hussain, J. (2023). Dependable intrusion detection system using deep convolutional neural network: A novel framework and performance evaluation approach. Telematics and Informatics Reports, 11, 100077.
[11] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765-4774..
[12] McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Transactions on Information and System Security (TISSEC), 3(4), 262-294.
[13] Al-Hitawi MAS and Máté GN. Enhancing Transformer-Based Language Models for Hungarian Handwritten Text Recognition [version 1; peer review: 1 approved with reservations]. F1000Research 2026, 15:181 (https://doi.org/10.12688/f1000research.176408.1).
[14] Padhiar, S., & Patel, R. (2023). Outside the closed world: On using machine learning for network intrusion detection. In International Conference on Information and Communication Technology for Intelligent Systems (pp. 265-270). Singapore: Springer Nature Singapore.
[15] Revathi, S., & Malathi, A. (2013). A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. International Journal of Engineering Research & Technology (IJERT), 2(12), 1848-1853.
[16] Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(5), 649-659.
[17] N. A. Mohammed et al., "Recognizing Phishing in Emails by Using Natural Language Processing and Machine Learning Techniques," Proc. 3rd Int. Conf. on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2025, pp. 1–7, doi: 10.1109/ICCR67387.2025.11292212.

BIOGRAPHIES OF AUTHORS

Assist. Lect. Marwan Ghazwan Mitab received his M.Sc. degree in Artificial Intelligence from Arab Amman University, Jordan, in 2022. Worked at Al-Nahrain University from 2009 to 2024.

Currently he is working as a lecturer at the University of Fallujah / College of Information Technology, and my research interests include artificial intelligence, machine learning, and intelligent systems.

Correspondence email: marwan.ghazwan@uofallujah.edu.iq



Assist. Lect. Raad Maklef Khalefa, Received His MSc. degree in the Information System from Osmania University – India in 2013.

He has been a full-time lecturer in College of Information Technology - University of Fallujah, Al-Anbar, Iraq, since March 2013. ongoing He can be contacted at email: raadmuaklef78@uofallojah.edu.iq.

Dr. Senan Ali Abd is a Lecturer in the Department of Cybersecurity, College of Information Technology, University of Fallujah, Anbar, Iraq.

He is actively involved in teaching and academic activities in the field of cybersecurity. His research interests include information security, network security, and emerging cybersecurity technologies. He can be contacted at senan.a.abd@uofallujah.edu.iq.