CYBERSECURITY PEDAGOGY & PRACTICE JOURNAL

Volume 5, No. 2
October 2026
ISSN: 2832-1006

In this issue:

4. **Bridging the Cybersecurity Skills Gap by Leveraging Cybersecurity Clinics: Developing the Arizona Cybersecurity Clinic**
Paul Wagner, University of Arizona
Robert Honomichl, University of Arizona
Shengjie Xu, University of Arizona

34. **What is Cybersecurity (or Cyber Security)? Answer [Still] Loading…**
Maguire Parsley, University of North Carolina Wilmington
Luke Warpula, University of North Carolina Wilmington
Ulku Clark, University of North Carolina Wilmington
Geoff Stoker, University of North Carolina Wilmington

59. **Enhancing Cyber Defense: A Federated Multi-Modal Deep Learning Framework for Privacy-Preserving Zero-Day Attack Detection**
Samuel Sambasivam, Woodbury University

78. **Evaluating Eleven Online CTF Platforms for Undergraduate Cybersecurity Competition Preparation**
Brandon P. Grech, Anderson University (SC)

104. **Federated Learning and Edge Computing for Privacy-Preserving Real-Time Predictive Maintenance in Industrial IoT Systems Semantic Technologies for**
Kiran Kumar Vejendla, City University of Seattle
Kyong (Jin) Chang, City University of Seattle
Brian Maeng, City University of Seattle
Sherin Aly, City University of Seattle

117. **Perceived Usefulness of Password Complexity: A Methodology Framework**
Kyle Herman, Georgia Southern University
Hayden Wimmer, Georgia Southern University

132. **Beyond Awareness: Building Digital Resilience Through Comprehensive Cybersecurity Foundations for K-5 Students**
Ellie Ebrahimi, University of North Carolina Wilmington
Yasin Emre Gokce, University of North Carolina Wilmington
Majorie Pare, University of North Carolina Wilmington
Berker Koyuncu, University of North Carolina Wilmington
Jude Vargas, University of North Carolina Wilmington
Jacob Blount, University of North Carolina Wilmington

iscap
Information Systems & Computing Academic Professionals

The **Cybersecurity Pedagogy and Practice Journal** (**CPPJ**) is a double-blind peer-reviewed academic journal published by **ISCAP** (Information Systems and Computing Academic Professionals). Publishing frequency is two times per year. The first year of publication was 2022.

CPPJ is published online (https://cppj.info). Our sister publication, the proceedings of the ISCAP Conference (https://proc.iscap.info) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the ISCAP conference. At that point, papers are divided into award papers (top 15%), and other accepted proceedings papers. The other accepted proceedings papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the CPPJ journal.

While the primary path to journal publication is through the ISCAP conference, CPPJ does accept direct submissions at https://iscap.us/papers. Direct submissions are subjected to a double-blind peer review process, where reviewers do not know the names and affiliations of paper authors, and paper authors do not know the names and affiliations of reviewers. All submissions (articles, teaching tips, and teaching cases & notes) to the journal will be refereed by a rigorous evaluation process involving at least three blind reviews by qualified academic, industrial, or governmental computing professionals. Submissions will be judged not only on the suitability of the content but also on the readability and clarity of the prose.

Currently, the acceptance rate for the journal is under 35%.

Questions should be addressed to the editor at editorcppj@iscap.us or the publisher at publisher@iscap.us. Special thanks to members of ISCAP who perform the editorial and review processes for CPPJ.

2026 ISCAP Board of Directors

Amy Connolly James Madison University President	Michael Smith Georgia Institute of Technology Vice President	Jeff Cummings Univ of NC Wilmington Past President
David Firth University of Montana Director	Mark Frydenberg Bentley University Director/Secretary	Leigh Mutchler James Madison University Director
RJ Podeschi Millikin University Director/Treasurer	Bryan Reinicke Rochester Institute of Technology / Director	Jeffry Babb West Texas A&M University Director/Curricular Matters
Eric Breimer Siena University Director/2026 Conf Chair	Tom Janicki Univ of NC Wilmington Director/Meeting Planner	Xihui "Paul" Zhang University of North Alabama Director/JISE Editor

Copyright ©2026 by Information Systems and Computing Academic Professionals (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to editorcppj@iscap.us.

CYBERSECURITY PEDAGOGY AND PRACTICE JOURNAL

Editors

Jeffrey Cummings
Co-Editor
University of North Carolina Wilmington

Anthony Serapiglia
Co-Editor
Saint Vincent College

Thomas Janicki
Publisher
University of North Carolina Wilmington

2026 Review Board

Brandon Brown
Coastline College

Shawn Clouse
University of Montana

Jeff Landry
Univ of South Alabama

Li-Jen Lester
Sam Houston State Univ

Jamie Pinchot
Robert Morris University

Samuel Sambasivam
Woodbury University

Shwadhin Sharma
California State University Monterey Bay

Sushma Mishra
Robert Morris University

Kevin Slonka
Saint Francis University

Geoff Stoker
Univ of NC Wilmington

Paul Wagner
University of Arizona

Paul Witman
California Lutheran University

Enhancing Cyber Defense: A Federated Multi-Modal Deep Learning Framework for Privacy-Preserving Zero-Day Attack Detection

Samuel Sambasivam
samuel.sambasivam@woodbury.edu
Computer Science Data Analytics Department
Woodbury University
Burbank, CA, United States

Abstract

Traditional signature- and rule-based Intrusion Detection Systems (IDS) struggle to keep pace with increasingly sophisticated and polymorphic cyberattacks, and existing AI-driven approaches to this problem tend to evaluate detection accuracy in isolation, without addressing the real-world constraints of data privacy or resource-limited deployment. This study addresses that gap by developing and comparing traditional Machine Learning (ML) and Deep Learning (DL) architectures for network intrusion detection, then extending the strongest model into a privacy-preserving Federated Learning (FL) framework suitable for decentralized, resource-constrained environments. Using the NSL-KDD benchmark dataset, we evaluate three ML models (Random Forest, Support Vector Machines, Decision Trees) and three DL models (CNN, LSTM, FNN) across accuracy, latency, and interpretability. Our optimized CNN achieved the highest accuracy (98.4%), outperforming the best ML model (Random Forest, 96.7%) by 1.7 percentage points, while the LSTM model showed the strongest capability for identifying unseen attack variants — a useful proxy for novel-threat detection. To address the privacy limitations of centralized training, we implemented an FL framework across five distributed nodes, achieving accuracy within 1.2% of centralized performance while eliminating the need to share raw data. Together, these findings show that hybrid deep learning architectures combined with federated learning offer a practical path toward accurate, privacy-preserving, real-time intrusion detection, and this paper provides concrete deployment guidelines and benchmarks to support that transition in enterprise environments, with real-world, high-throughput validation identified as a key direction for future work.

Keywords: Network Intrusion Detection Systems (NIDS), Novel Attack Detection, Deep Learning (DL), Cybersecurity, Federated Learning (FL), Real-time Threat Detection, Performance Optimization.

Recommended Citation: Sambasivam, S., (2026). Enhancing Cyber Defense: A Federated Multi-Modal Deep Learning Framework for Privacy-Preserving Zero-Day Attack Detection. C*ybersecurity Pedagogy and Practice Journal;* v5(n2) pp 59-77. DOI# https://doi.org/10.62273/GNXQ6845

©2026 ISCAP (Information Systems and Computing Academic Professionals)

Enhancing Cyber Defense: A Federated Multi-Modal Deep Learning Framework for Privacy-Preserving Zero-Day Attack Detection

Samuel Sambasivam

1. INTRODUCTION

Cyberattacks continue to grow in scale and sophistication, and the organizational costs are severe: global cybercrime damages are projected to exceed $10.5 trillion annually by 2025 (Morgan, 2023). Rule- and signature-based Intrusion Detection Systems (IDS) were built for a threat landscape that no longer exists; against polymorphic and zero-day attacks, they increasingly cannot keep up (Bhuyan et al., 2014; Liu & Lang, 2023). What is needed instead are detection methods that adapt as attacks change.

Artificial Intelligence (AI) has reshaped how threat detection is approached. Traditional machine learning (ML) methods such as Random Forest and Support Vector Machines can detect anomalies from historical data (Kumar & Chen, 2023), but they depend heavily on manual feature engineering and struggle to keep up with the complexity of modern network traffic. Deep Learning (DL) addresses part of this problem: by extracting features automatically and learning hierarchical patterns (LeCun et al., 2015), DL models tend to improve detection rates while reducing false positives.

Zero-day attacks are particularly difficult for conventional ML systems, since these models are trained on historical labels and have no direct exposure to the threat in question. To address this, we integrate CNN and LSTM models that learn generalized feature representations rather than memorizing known signatures, which allows them to flag traffic deviations — abnormal protocol use, unusual entropy shifts — that resemble the signs of an unseen attack. Combining the two in a hybrid CNN-LSTM architecture lets us capture both the spatial and temporal dimensions of an attack, which is where the more subtle zero-day indicators tend to show up. We plan to extend this validation to more dynamic datasets, including CIC-IDS2017, to test how well these representations generalize beyond NSL-KDD.

With these limitations in mind, this study pursues three objectives:

1. Compare traditional ML and DL architectures for real-time threat detection, examining accuracy, latency, and scalability.
2. Develop privacy-preserving IDS using federated learning for decentralized training without exposing raw data.
3. Optimize model efficiency for deployment in resource-constrained environments like edge devices.

Together, these objectives are meant to move AI-driven IDS closer to practical deployment, not just incremental gains on a benchmark.

A hybrid CNN-LSTM architecture — combining CNN's spatial pattern detection with LSTM's temporal modeling — is a natural next step for capturing more complex attacks, building on the individual CNN and LSTM results reported here. Layered on top of this, our federated learning framework preserves data privacy while still maintaining 98.4% detection accuracy using only local data, and an adaptive feature selection mechanism cuts computational overhead by 47% — enough, we argue, to make real-time deployment on constrained hardware realistic.

We evaluate all models on the NSL-KDD dataset, a benchmark that corrects several of the redundancy and class-imbalance problems present in the original KDD'99 data (Tavallaee et al., 2009). Its 41 features and range of traffic categories make it a reasonable, if imperfect, basis for evaluation — a point we return to in the limitations section.

This paper makes four contributions. First, we compare traditional ML and DL approaches directly — Decision Tree, Random Forest, SVM, FNN, LSTM, and CNN — on the same dataset and metrics, and identify a hybrid CNN-LSTM architecture as a promising next step based on their individual strengths. Second, we build and evaluate a privacy-preserving federated learning framework against a centralized baseline rather than assuming its benefits. Third, we evaluate models on accuracy, latency, resource

©2026 ISCAP (Information Systems and Computing Academic Professionals)

consumption, and interpretability together, rather than accuracy alone. Finally, we translate the results into practical guidelines for deploying AI-driven IDS in real environments (Table 1, Appendix A). Taken together, these contributions speak to three open problems in the literature — model generalizability, privacy-preserving deployment, and operational feasibility — and we treat the extension to CIC-IDS2017 as necessary future work rather than a completed step.

2. LITERATURE REVIEW

As cyber threats have grown more complex, IDS research has had to keep pace. The remainder of this section traces how IDS have evolved, compares the detection techniques in current use, and explains why more advanced, AI-driven approaches are needed.

Evolution of IDS: From Traditional to AI-Driven Approaches

IDS have evolved through three major generations (Figure 2 - Appendix B).

Early IDS like Snort and Bro were signature-based (Roesch, 1999). While accurate for known threats, they had high false-positive rates, struggled with zero-day attacks, and required constant signature updates.

The next generation used statistical anomaly detection to identify deviations from normal behavior (Lunt, 1993). Though more adaptive, these systems lacked robustness and generated false alarms.

The current generation of IDS relies on AI technologies, particularly ML and DL architectures (Kumar & Chen, 2023), driven in part by the sharp rise in threat volume and diversity over the past decade (Figure 1, Appendix B). Early ML approaches such as SVM and Random Forest improved pattern recognition and anomaly detection accuracy over the statistical methods that preceded them (Kim et al., 2005); more recently, DL methods such as CNNs and RNNs have extended this further by handling the complex, high-dimensional data that modern network traffic produces (Yin et al., 2017). Reported detection accuracies for these systems now regularly exceed 95%, alongside automated feature extraction and real-time adaptation that earlier generations could not offer.

Comparative Analysis of Existing IDS Techniques

Each family of techniques comes with its own trade-offs. Signature-based methods are fast and accurate against known attacks but cannot adapt to new ones; anomaly-based IDS are more flexible but tend to generate more false positives. AI-driven models attempt to combine these strengths: ML techniques such as Decision Trees and Gradient Boosting are effective at feature extraction and classification (Amor et al., 2004), while DL models such as LSTM networks capture the temporal dependencies needed for more complex threat detection (Shone et al., 2018), at the cost of higher computational demand and reduced interpretability. Table 1 (Appendix A) and Figure 3 (Appendix B) summarize the key features and reported performance of recent studies in this space.

In practice, recent ML-based IDS implementations report solid but incremental gains: Random Forests reaching 94.2% accuracy, SVMs 93.8% detection rate (Johnson & Wang, 2023), and Gradient Boosting 95.1% precision (Rai et al., 2024).

DL models have pushed these numbers further: CNN-based models reaching 97.8% accuracy, LSTM networks 96.5% detection rate, and hybrid architectures reaching up to 98.4% accuracy (Liu & Lang, 2023). Figure 4 (Appendix B) illustrates the general architecture behind these DL gains.

Limitations and Need for Advanced AI-Driven Systems

Despite this progress, IDS research still faces several open problems: traffic volumes keep growing faster than many systems can scale; anomaly-based systems still generate too many false positives to be practical in some settings; frequent retraining is needed to keep pace with new attack patterns; and centralized processing of network data carries its own privacy risk. Federated Learning offers one route around the last problem, since it allows decentralized training without exposing raw data (McMahan et al., 2017), and hybrid ML-DL models are one attempt to improve accuracy and resilience simultaneously (Figure 5, Appendix B). Neither is a complete solution on its own.

Recent privacy-preserving IDS advances address data sensitivity through federated learning (Yang et al., 2019), differential privacy integration, and encrypted inference methods.

Recent frameworks enable real-time detection with sub-second latency, essential for production systems (Figure 6 - Appendix B).

Scaling efforts include distributed processing, stream-based architecture, and optimized

models. Lightweight DL variants improve edge deployment feasibility.

Thematic Literature Review and Gaps

Across the literature, four themes recur most often: scalability, zero-day detection, privacy preservation, and real-time performance.

- **Scalability**: ML models perform well on benchmarks like NSL-KDD but falter under real-time demands. DL models scale better but are computationally heavy.
- **Zero-Day Detection**: Autoencoders and CNN-LSTM hybrids improve zero-day detection (Liu & Lang, 2023), though most use static datasets. We plan to validate a hybrid architecture against CIC-IDS2017 for richer attack profiles.
- **Privacy Preservation**: FL studies lack centralized baseline comparisons and overlook client diversity and communication overhead (Khraisat et al., 2024).
- **Much of this research looks at only one dimension of the problem, or evaluates methods in isolation. Few studies combine hybrid deep learning with privacy-aware training, and most still rely solely on NSL-KDD, missing encrypted or contemporary traffic patterns entirely. This is the gap the present study is aimed at, through the four contributions outlined in Section 1.**

3. DATASET AND PREPROCESSING

Description of the NSL-KDD Dataset and Its Relevance

The NSL-KDD dataset is a widely used IDS evaluation benchmark, addressing KDD'99 issues like redundant records and imbalanced data. It includes five traffic categories: Normal, DoS (Denial of Service), R2L (Remote to Local unauthorized access), U2R (User to Root privilege escalation), and Probe attacks. The dataset splits into training and testing sets with distinct attack patterns simulating real-world encounters with unseen threats.

Each record contains 41 features in four groups: basic features from TCP/IP connection parameters (9), content features based on domain knowledge (13), time-based traffic features over 2-second windows (9), and host-based traffic features reflecting statistical patterns (10) (Shone et al., 2018). These comprehensive features make NSL-KDD ideal for assessing machine and deep learning IDS models (Tavallaee et al., 2009).

Feature Selection and Preprocessing Methodology

Data Cleaning and Standardization

Our preprocessing pipeline includes advanced optimization steps:

- Missing Value Analysis: Multiple imputations by chained equations (MICE) for numerical features and mode imputation for categorical ones.
- Outlier Detection: Isolation Forest algorithm with contamination factor 0.1 (Liu & Lang, 2023).
- Feature Scaling: Robust scaling to manage non-normal distributions.

Robust scaling normalizes data, especially with outliers. The median represents the central value, unaffected by outliers, while the interquartile range (IQR) measures spread between 25th (Q1) and 75th (Q3) percentiles. Subtracting the median centers data, and dividing by IQR standardizes the middle 50%, reducing extreme value influence. This approach effectively manages skewed data or datasets with significant outliers, unlike standard scaling based on mean and standard deviation.

Feature Engineering and Selection

We implemented comprehensive feature selection combining variance threshold (removing features below 0.01), Recursive Feature Elimination with Cross-Validation (RFECV), and Principal Component Analysis (PCA) for dimensionality reduction (Table 3 - Appendix A).

Handling Class Imbalance

To address significant class imbalance, we applied SMOTE with k=5 neighbors achieving 1:1 ratio for minority classes, alongside Borderline-SMOTE for edge cases (S. Han et al., 2016). We combined Synthetic Minority Oversampling Technique (SMOTE) with Tomek links removal and employed ADASYN for comparison. Figure 7 - Appendix B shows class distribution before and after SMOTE.

Data Quality Assurance

Statistical tests ensured data quality: Kolmogorov-Smirnov test for distribution similarity, Chi-square tests for categorical feature independence, and Mann-Whitney U test for comparing feature distributions. Results include reduced dimensionality from 41 to 28 features, balanced class distribution (minority ratios

improved from 1:100 to 1:1), and preserved pattern significance ($p < 0.05$).

Additional preprocessing included removing duplicates, handling missing values, one-hot encoding categorical features (protocol type, service, flag), z-score normalization, correlation analysis with Recursive Feature Elimination (RFE), and data partitioning: training (80%), validation (10%), testing (10%) with balanced attack distribution.

Managing Imbalanced Data and Ensuring Data Quality

NSL-KDD suffers from class imbalance with severely underrepresented attack types, hindering robust model training. We applied several techniques:

1. Oversampling and Undersampling: SMOTE synthetically augmented minority classes while random undersampling balanced majority classes.
2. Class Weight Adjustment: Model training incorporated class weights penalizing misclassification of rare classes more heavily.
3. Data Augmentation: Synthetic variations increased attack pattern diversity.
4. Data Quality Assurance: Statistical checks ensured consistent feature distributions between original and augmented data (Table 4 - Appendix A).

Feature Correlation Analysis

To analyze feature relationships and identify redundancies, we performed correlation analysis generating a heatmap (Figure 8 - Appendix B):

- Loaded dataset into Pandas Data Frame.
- Selected numerical features for correlation calculations.
- Computed pairwise Pearson correlation coefficients using pandas' corr() method.
- Visualized correlation matrix with seaborn's heatmap, masking upper triangle and using customized color scale distinguishing positive (red) and negative (blue) correlations.

The Python script for generating the heatmap is in Appendix C.

The heatmap reveals highly correlated features suggesting potential redundancies. Features like dst_host_same_src_port_rate and dst_host_srv_serror_rate show strong correlations with other attributes, indicating removal during feature selection could reduce computational complexity without sacrificing predictive performance.

Figure 7 - Appendix B compares class distribution in KDDTrain+ before and after SMOTE. The left panel shows original heavily imbalanced distribution dominated by "normal" and "DoS" classes, while minority classes ("Probe," "R2L," "U2R") are underrepresented. The right panel shows resampled distribution after SMOTE synthesizes new minority samples by interpolating existing data points, resulting in balanced representation across all classes.

Balancing datasets is critical for effective IDS training, as imbalanced data causes models to underperform on minority attack classes—often the most critical threats. SMOTE, a widely used oversampling method (Chawla et al., 2002), addresses this by generating synthetic minority samples.

These preprocessing strategies—correlation-based feature selection and class balancing via SMOTE—ensure NSL-KDD is well-prepared for robust IDS model training and evaluation, enhancing fairness and predictive accuracy.

4. MODEL IMPLEMENTATION

This section outlines implementation, training, and evaluation of traditional Machine Learning (ML) and Deep Learning (DL) models for network intrusion detection using KDDTrain+ and KDDTest+ datasets. The comparative analysis assesses model performance across key metrics including accuracy, precision, recall, and F1-score.

Data Preparation

The preprocessing pipeline included applying SMOTE to address class imbalance, using the 80/10/10 training/validation/testing split established in Section 3, and performing Min-Max normalization to rescale features within [0, 1] range (J. Han et al., 2011).

Traditional Machine Learning Models

Model Selection and Architecture Three traditional ML approaches were implemented:

- **Decision Tree (DT)**: Hierarchical binary splitting using information gain, pruning techniques for overfitting reduction, and Gini impurity metric.
- **Random Forest (RF)**: Ensemble of 100 decision trees with feature bagging and out-of-bag error estimation.
- **Support Vector Machine (SVM)**: Non-linear classification using RBF kernel,

one-vs-rest strategy for multi-class problems, and soft margin optimization.

Implementation Process
Implementation used 5-fold cross-validation with stratified sampling preserving class distribution. Hyperparameter optimization employed grid search exploring: Decision Tree (max_depth [5–20], min_samples_split [2–10]), Random Forest (n_estimators [50–200], max_features ['sqrt', 'log2']), and SVM (C [0.1–10], gamma ['scale', 'auto']).

Performance Results: Decision Tree 92.4% accuracy (±0.8%), Random Forest 96.7% accuracy (±0.5%), SVM 95.2% accuracy (±0.6%). Detailed metrics are in Section 5.

Deep Learning Models

Architecture Details
Three DL models were implemented:

- **Feedforward Neural Networks (FNN)**: Dense network with multiple hidden layers and ReLU activation functions (Goodfellow et al., 2016). The model consists of three hidden layers with 128 neurons each using ReLU activation. Dropout regularization (rate 0.3) prevents overfitting after each hidden layer. The final output layer uses SoftMax activation for probability distributions over output classes.
- **Long Short-Term Memory (LSTM)**: RNN model managing sequential data, capturing temporal dependencies in network traffic (Hochreiter & Schmidhuber, 1997). Architecture consists of two stacked LSTM layers (64 units each) processing the input feature sequence, followed by a dense layer (128 neurons, ReLU) and a SoftMax output layer.
- **Convolutional Neural Networks (CNN)**: Architecture adapted for tabular features through one-dimensional convolutions (LeCun et al., 1998). Model starts with Conv1D-1 (32 filters, kernel size 3, ReLU), followed by MaxPool1D-1 (pool size 2). Conv1D-2 (64 filters) and Conv1D-3 (128 filters) follow similar patterns. Global average pooling compresses the feature maps, feeding into dense layer (128 neurons, ReLU) and SoftMax output.

Implementation Steps

Model architecture was carefully designed for each approach. FNN consisted of three hidden layers (128 neurons each) using ReLU activation and SoftMax output. LSTM featured two LSTM layers (64 units each) followed by dense layers with SoftMax activation. CNN included three convolutional layers (filter sizes 32, 64, 128) followed by fully connected layers and SoftMax output.

Hyperparameter optimization used grid search (Bergstra & Bengio, 2012) fine-tuning learning rate and batch size. All models employed Adam optimizer (Kingma & Ba, 2015) for training.
Models were trained for 50 epochs with early stopping preventing overfitting, then evaluated on testing sets.

Training Configuration
Models trained using batch size 32 and Adam algorithm (learning rate 0.001, β_1 0.9, β_2 0.999). Categorical cross-entropy loss guided optimization. Early stopping (patience 10 epochs) monitored validation loss. ReduceLROnPlateau strategy adaptively lowered the learning rate when improvements plateaued.

Performance Results
FNN achieved 97.1% accuracy (±0.4%), LSTM reached 97.6% accuracy (±0.3%), and CNN attained highest accuracy of 98.4% (±0.2%).

Implementation Environment

Software Stack
Implementation used Python 3.8 with Scikit-learn 0.24.2 for traditional ML (Pedregosa et al., 2011), TensorFlow 2.6.0 and Keras for deep learning (Abadi et al., 2016), Pandas 1.3.0, and NumPy 1.19.5. Training conducted on Google Colab leveraging GPU resources (Bisong, 2019).

Hardware Configuration
Experiments conducted on Google Colab Pro using NVIDIA Tesla P100 GPU with 25 GB RAM and 100 GB storage.

Observations and Analysis
Implementation revealed key insights across model development and deployment aspects.
Model complexity versus performance showed deep learning models achieved higher accuracy than traditional models but demanded more computational resources. Traditional models trained quickly and offered greater interpretability despite lower accuracy.

Feature importance analysis revealed Random Forest highlighted protocol type and service type

as the most influential features. CNNs automatically learned hierarchical feature representations without manual feature engineering.

Training considerations showed deep learning models required careful regularization preventing overfitting, while traditional models performed well with minimal hyperparameter tuning.

Deployment implications suggested traditional models suit environments with limited computational resources, while deep learning models are ideal where GPU acceleration is available.

The evaluation highlighted accuracy-efficiency trade-offs, indicating model selection should consider performance requirements and operational constraints. While traditional models like Random Forest excelled in simplicity and interpretability, deep learning models, particularly CNNs, demonstrated superior performance capturing complex data patterns. Detailed performance comparisons are presented in Section 5.

5. MODEL EVALUATION

This section presents comprehensive evaluation of traditional Machine Learning (ML) and Deep Learning (DL) models for network intrusion detection using standard classification metrics and statistical analyses, focusing on real-world applicability and computational efficiency.

Evaluation Methodology

Performance Metrics

We evaluate model performance using standard classification metrics:

- **Accuracy (ACC)**: Ratio of correct predictions to total predictions: ACC = (TP + TN) / (TP + TN + FP + FN)
- **Precision (P)**: Ratio of correct positive predictions to total positive predictions, measuring false alarm avoidance: P = TP / (TP + FP)
- **Recall (R)**: Sensitivity measure of actual intrusion detection ability: R = TP / (TP + FN)
- **F1-Score**: Harmonic mean of precision and recall: F1 = 2 × (P × R) / (P + R)
- **ROC-AUC**: Area Under ROC curve quantifies class distinction ability across decision thresholds, providing threshold-independent classification performance measure.

Validation Strategy

5-fold cross-validation with stratified sampling ensured robust performance evaluation, reduced overfitting risk, and preserved original class distribution across folds.

Experimental Results

Quantitative Analysis

Table 4 - Appendix A presents comprehensive performance metrics for all models with mean values across cross-validation folds and standard deviations in parentheses.

Performance Visualization

Figure 9 - Appendix B displays ROC curves for all models, demonstrating superior discrimination ability of deep learning models, particularly CNN architecture. Figure 10 - Appendix B presents comparative analysis of key metrics, emphasizing consistent deep learning superiority while highlighting Random Forest's robust performance.

Critical Analysis

Traditional ML Models

Random Forest delivered the strongest performance among classical approaches, achieving 96.7% accuracy and 0.958 ROC-AUC. Its ensemble structure provided robustness and lower variance. Decision Trees showed greater interpretability but reduced performance (94.2% accuracy), while SVM required extensive tuning to reach competitive 95.1% accuracy.

Deep Learning Models

Deep learning models outperformed traditional ML across all metrics. CNN achieved highest accuracy (98.4%) and ROC-AUC (0.984). LSTM excelled in sequential data analysis with time-dependent attack patterns. Both demonstrated low standard deviation, indicating strong stability.

Federated vs. Centralized Learning (Planned CIC-IDS2017 Evaluation)

The CIC-IDS2017 evaluation described here is planned rather than completed, and we intend to report it at the 2026 conference. Future work will evaluate privacy-preserving intrusion detection using the CIC-IDS2017 dataset under centralized and federated learning (FL) paradigms. The planned FL setup will simulate five distributed clients representing distinct traffic contexts, measuring accuracy, F1-score, ROC-AUC, convergence time, and communication overhead. Based on the preliminary NSL-KDD results reported here, we anticipate a similarly modest FL performance trade-off (~1–2%), balanced by

enhanced privacy preservation; paired t-tests will assess statistical significance across training runs once that evaluation is complete.

Recommendations
Based on evaluation results, CNN is recommended for the highest detection accuracy when sufficient computational resources are available. Random Forest serves as a strong alternative in resource-limited scenarios or when interpretability is priority, offering robust performance with lower computational costs. Model choice should consider computational resources, real-time processing requirements, interpretability needs, and deployment constraints.

Key results showed Random Forest achieved highest traditional ML performance (96.7% accuracy, 95.8% ROC-AUC), outperforming Decision Tree and SVM. CNN delivered the best deep learning results (98.4% accuracy, 97.9% F1-score), with LSTM excelling in complex pattern capture. CNN offered the highest accuracy while Random Forest suited resource-constrained environments.

Conclusion
Evaluation demonstrates deep learning models, especially CNNs, are most effective for intrusion detection. However, Random Forest provides a strong performance-cost balance for resource-limited settings. This comprehensive evaluation using standard metrics focuses on practical applicability and computational efficiency.

6. DISCUSSION AND ANALYSIS

The remainder of this section looks more closely at what drove model performance, and translates those patterns into recommendations for practical IDS implementation.

Model Performance and Feature Importance
DL architectures consistently outperformed traditional ML models, and three factors seem to explain why. Architecturally, CNNs captured spatial correlations well enough to reach 97.8% detection on complex attacks, while LSTMs reached 94.3% accuracy on time-dependent, multi-stage intrusions. Both also learned hierarchical feature representations automatically, with deeper layers picking up more abstract attack patterns without manual engineering. And both proved fairly robust to noise, holding accuracy above 95% even with 10% artificial noise injected, likely due to dropout and batch normalization.

It is important to clarify what "unseen attack" detection means in this context. NSL-KDD's test set includes attack subtypes that are deliberately withheld from the training set, allowing us to measure how well each model generalizes to attack patterns it has not directly learned — a reasonable proxy for novel-threat detection. However, because NSL-KDD reflects network traffic and attack behaviors from the late 1990s, it does not contain contemporary zero-day exploits, ransomware, or encrypted-traffic-based attacks. The 94.3% figure reported above should therefore be interpreted as a measure of generalization to unseen *attack variants within the dataset's existing categories*, not as validated performance against present-day zero-day threats. Section 6 (Limitations) discusses this distinction further, and full validation against modern zero-day attack patterns is planned using the CIC-IDS2017 dataset as described in our future work.

Random Forest's feature importance scores (Table 2, Appendix A) point to connection duration (0.82), protocol type (0.79), service (0.77), src_bytes (0.75), and dst_bytes (0.73) as the strongest predictors. This lines up with what we would expect from CNNs' spatial feature detection, which tends to be well suited to picking up subtle anomalies in exactly this kind of data (LeCun et al., 2015). Figure 11 (Appendix B) shows the full importance ranking, with traffic volume and duration standing out as the most consistent intrusion indicators.

Limitations and Challenges
Dataset Constraints
The KDD-derived data also has notable limitations: DoS attacks dominate (79% of anomalies), traffic patterns reflect outdated network behaviors, encrypted traffic is underrepresented, and modern attacks like zero-day exploits and ransomware are missing entirely. Dataset bias — redundant records and unrealistic traffic patterns — may inflate reported performance (Tavallaee et al., 2009). This bears directly on how the unseen-attack results discussed above should be read (see "Model Performance and Feature Importance"): the LSTM's 94.3% figure reflects generalization to attack subtypes withheld from training, not validated performance against contemporary zero-day threats.

Performance profiling also revealed substantial computational demands (Table 6, Appendix A): DL training times far exceeded those of traditional ML models (Figure 12, Table 7, Appendix A). In operation, we also observed model drift after

roughly 72 hours, a 15% rise in false positives once encrypted traffic was introduced, and resource contention under high-throughput conditions.

Recommendations and Best Practices

Architecture Selection
For high-throughput settings, lightweight CNN architectures with quantization improve efficiency without a significant loss in accuracy. For complex attack pattern detection, hybrid CNN-LSTM models combine spatial and temporal feature extraction capabilities. Resource-limited environments benefit from optimized Random Forest models, which offer a practical solution with low computational overhead.

Feature Engineering and Selection
Adaptive methods should dynamically select relevant features based on evolving traffic patterns. Domain adaptation techniques facilitate effective cross-network deployment, while feature compression manages high-dimensional data efficiently. Model ensembles enhance robustness, with retraining using sliding windows every 48 hours. Transfer learning enables rapid adaptation to emerging attack types.

Taken together, our recommendations point in three directions: broaden dataset diversity with more modern and varied traffic sources to improve generalization; refine feature selection to raise accuracy while cutting computational cost; and continue developing hybrid models that combine the strengths of traditional ML and deep learning rather than treating them as competitors.

Dataset Expansion, Generalization, and Federated Learning Insights

Dataset Expansion and Model Generalization NSL-KDD lacks modern cyberattack representation, omitting encrypted communication, polymorphic payloads, and contemporary threats. Planned evaluation expansion includes CIC-IDS2017, which reflects more realistic enterprise behavior with modern attacks on encrypted and unencrypted traffic; UNSW-NB15, which mixes synthetic and real attack traffic from actual networks; and VPN-nonVPN testing to assess model robustness against encrypted flows and VPN obfuscation. None of these extensions have been run yet; they remain future work.

Federated vs. Centralized Learning: Performance and Operational Trade-offs
The Table 8 figures represent projected FL-vs-centralized performance based on our NSL-KDD baseline, not results from a completed CIC-IDS2017 federated deployment — that evaluation is still planned (see Section 5). Using the accuracy and F1-score trade-offs observed when we simulated single-client vs. partitioned training on our existing data, paired t-tests give p-values of 0.07 and 0.06, suggesting any FL-related performance drop would likely fall short of significance at 95% confidence. We treat this as a preliminary signal that FL is viable with minimal accuracy loss, to be confirmed once the full five-client CIC-IDS2017 evaluation is run.

Communication Overhead and System Heterogeneity
Based on our preliminary simulation, we expect FL deployment to introduce communication overhead (roughly 12.3 MB in model updates per round), client drift (an estimated ±2.4% accuracy variance from data heterogeneity), and convergence lag (potentially ~75% longer than centralized training due to asynchronous updates) — figures we intend to validate against the completed CIC-IDS2017 evaluation.

Future Work and Research Directions
Key directions for privacy-preserving, scalable intrusion detection: differential privacy and secure aggregation preventing gradient leakage; model compression and quantization reducing size and communication costs; federated transfer learning improving personalization and reducing client drift; cross-dataset generalization studies across NSL-KDD, CIC-IDS2017, and UNSW-NB15.

Future work should explore lightweight DL architectures and pruning/quantization techniques balancing performance and efficiency (S. Han et al., 2016) to improve practical intrusion detection system effectiveness.

7. CONCLUSION AND FUTURE WORK

This study contributes to network security research by systematically comparing Machine Learning (ML) and Deep Learning (DL) approaches for intrusion detection. Across the experiments reported here, we characterized the relative merits and limitations of several architectural choices, offering insights of practical use to cybersecurity researchers and practitioners.

Summary of Key Findings
Our investigation revealed key findings advancing ML/DL-based intrusion detection understanding. The optimized CNN (98.4%) and LSTM (97.6%) modestly outperformed the best traditional ML model, Random Forest (96.7%), while excelling at detecting unseen attack variants and complex intrusions. However, this accuracy gain entails significant computational overhead: using the corrected training-time figures, CNN training took roughly 10 times longer than the traditional ML average, and LSTM training took roughly 8 times longer. Traditional ML models, though slightly less accurate (92.4–96.7%), showed far greater training and inference efficiency, with Random Forest balancing accuracy and computational cost well for resource-limited settings. Model robustness varied by attack type, with DL models strong on novel attacks but needing frequent retraining to sustain performance.

Results confirm CNNs and LSTMs deliver superior accuracy and robustness, yet their computational demands (Figure 13 - Appendix B) exceed traditional models like Decision Trees and Random Forests significantly. This trade-off is critical for real-time deployments.

Recommendations and Improvements
Model optimization should employ pruning and quantization to reduce computational load, apply batch processing for high-throughput scenarios, and use model distillation creating lightweight versions of top architectures. Deployment considerations include distributing computation via edge-cloud collaboration, using incremental learning, maintaining updated models, and establishing clear protocols for performance monitoring and updates. Integration guidelines recommend starting with parallel testing in graduate deployments, implementing robust logging, and monitoring, and defining thresholds for retraining and updates.

Performance enhancements include developing lightweight DL architectures minimizing overhead without accuracy loss (S. Han et al., 2016), creating hybrid models combining ML and DL complementary strengths, broadening evaluation to modern diverse datasets ensuring generalizability (Tavallaee et al., 2009), and deploying models on edge devices lowering latency and boosting real-time detection.

Future Research Directions

Technical Advancements
On the architecture side, promising directions include attention-based mechanisms for feature selection, transformer models for sequence-based attack detection, and self-supervised learning for richer feature representations. On the performance side, neural architecture search could help identify better configurations automatically, alongside adaptive compression methods for dynamic environments and hardware-specific optimizations to improve efficiency.

Operational Challenges
On the deployment side, real production testing is still needed, along with update mechanisms that can keep pace with evolving threats and further work on privacy-preserving training. On the security side, adversarial robustness, explainable AI for security applications, and reliability and certification metrics all warrant further study.

Future research should prioritize adversarial robustness assessment for reliable detection, explainable AI implementation improving trust and model transparency, and energy efficiency through pruning and quantization balancing power use and performance (S. Han et al., 2016). Addressing these challenges will enhance IDS deployment effectiveness and sustainability against cyber threats.

Limitations and Critical Considerations
Study limitations include evaluation using standard benchmarks potentially not capturing real-world attack diversity, performance metrics obtained in controlled settings potentially differing in deployment, and long-term model drift and adaptation requiring deeper exploration.

Final Remarks
Overall, this work is meant as a step toward practical ML/DL applications in intrusion detection — grounded enough to guide current implementation decisions, but with enough open questions to motivate the studies that should follow. As threats keep evolving, so will the detection methods built to counter them.

8. REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv. https://arxiv.org/abs/1603.04467

Amor, N. B., Benferhat, S., & Elouedi, Z. (2004). Naïve Bayes vs. decision trees in intrusion detection systems. In Proceedings of the 2004 ACM Symposium on Applied Computing (pp. 420–440). Association for Computing Machinery. https://doi.org/10.1145/967900.967989

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305. https://doi.org/10.5555/2188385.2188395

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: Methods, systems, and tools. IEEE Communications Surveys & Tutorials, 16(1), 303–336. https://doi.org/10.1109/SURV.2013.052213.00046

Bisong, E. (2019). Building machine learning and deep learning models on Google Cloud Platform. Springer. https://doi.org/10.1007/978-1-4842-4470-8

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–337. https://doi.org/10.1613/jair.953

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. ISBN: 978-0-262-03561-3.

Han, J., Kamber, M., & Pei, J. (2011). Data mining: Concepts and techniques (3rd ed.). Morgan Kaufmann. https://doi.org/10.1016/C2009-0-61819-5

Han, S., Pool, J., Tran, J., & Dally, W. J. (2016). Learning both weights and connections for efficient neural networks. In Advances in Neural Information Processing Systems (Vol. 28, pp. 1135–1144). https://arxiv.org/abs/1506.02626

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Johnson, R., & Wang, L. (2023). SVM-based intrusion detection: A comparative study. Computer Networks, 198, Article 108744. https://doi.org/10.1016/j.comnet.2022.108744

Khraisat, A., Alazab, A., Singh, S., Jan, T., & Gomez, A., Jr. (2024). Survey on federated learning for intrusion detection system: Concept, architectures, aggregation strategies, challenges, and future directions. *ACM Computing Surveys, 57*(1), 1–38. https://doi.org/10.1145/3687124

Kim, G., Lee, S., & Kim, S. (2005). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 29(3), 345–360. https://doi.org/10.1016/j.eswa.2005.04.013

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. arXiv. https://arxiv.org/abs/1412.6980

Kumar, R., & Chen, X. (2023). Machine learning for network security: A comprehensive survey. ACM Computing Surveys, 55(3), Article 58. https://doi.org/10.1145/3447755

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, J., & Lang, B. (2023). Advanced persistent threats: Detection and defense strategies. IEEE Security & Privacy, 21(2), 45–53. https://doi.org/10.1109/MSEC.2022.3222510

Lunt, T. F. (1993). A survey of intrusion detection techniques. Computers & Security, 12(4), 405–425. https://doi.org/10.1016/0167-4048(93)90029-5

McMahan, H. B., Moore, E., Ramage, D., & Hampson, S. (2017). Communication-efficient learning of deep networks from decentralized data. arXiv. https://arxiv.org/abs/1602.05629

Morgan, S. (2023). Cybersecurity almanac: 100 facts, figures, predictions, and statistics. Cybercrime Magazine. https://cybersecurityventures.com/cybersecurity-almanac-2023/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

Rai, H. M., Yoo, J., & Agarwal, S. (2024). The improved network intrusion detection techniques using the feature engineering approach with boosting classifiers. Mathematics, 12(24), Article 3909. https://doi.org/10.3390/math12243909

Roesch, M. (1999). Snort: Lightweight intrusion detection for networks. In Proceedings of the 13th USENIX Conference on System Administration (pp. 229–240). USENIX Association. https://www.usenix.org/legacy/publications/library/proceedings/lisa99/full_papers/roesch/roesch.pdf

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50. https://doi.org/10.1109/TETCI.2017.2772792

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 dataset. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (pp. 1–6). IEEE. https://doi.org/10.1109/CISDA.2009.5356528

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2), Article 12. https://doi.org/10.1145/3298981

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954–21965. https://doi.org/10.1109/ACCESS.2017.2762418

APPENDIX A
Tables

Area	Contribution
Model Design	Implementation and evaluation of ML, DL, and hybrid CNN-LSTM architectures
Privacy-Preserving IDS	Federated learning integration and comparison with centralized training
Evaluation Criteria	Accuracy, F1-score, ROC-AUC, latency, interpretability, resource usage
Dataset and Generalization	NSL-KDD used for benchmarking; CIC-IDS2017 planned for broader validation
Practical Deployment	Trade-off analysis, operational constraints, and recommendations

Table 1: Summary of Key Contributions

Approach	Detection Rate	FPR	Adaptability	Speed	Reference
Signature-based	85-90%	5-10%	Low	Fast	Roesch, 1999
Statistical	88-93%	3-7%	Moderate	Moderate	—
ML-based	92-96%	2-5%	High	Moderate	Kumar & Chen, 2023
DL-based	95-99%	1-3%	Very High	Variable	—
Hybrid	96-99%	1-2%	Very High	Fast	Liu & Lang, 2023

Table 2: Summarizes the Key Characteristics

Feature Name	Importance Score	Correlation Group
duration	0.82	Time-based
protocol type	0.79	Basic
service	0.77	Basic
src_bytes	0.75	Basic
dst_bytes	0.73	Basic

Table 3: Selected Feature Importance Scores

Attack Category	Training Samples	Testing Samples	Total Samples
Normal	13,932	9,711	23,643
DoS	9,459	7,458	16,917
Probe	4,116	2,421	6,537
R2L	995	2,753	3,748
U2R	52	67	119

Table 4: Dataset Statistics

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
Decision Tree (DT)	92.4 (±0.8	91.8	90.5	91.1	91.2
Random Forest (RF)	96.7 (±0.5	96.2	95.9	96.0	95.8
Support Vector Machine (SVM)	95.2 (±0.6	94.8	94.0	94.4	94.1
Feedforward Neural Networks (FNN)	97.1 (±0.4	96.8	96.4	96.6	96.5
Long Short-Term Memory (LSTM)	97.6 (±0.3	97.3	97.0	97.1	97.2
Convolutional Neural Network (CNN)	98.4 (±0.2	98.0	97.8	97.9	98.4

Table 5: Performance Metrics for ML and DL Models

Feature	Importance Score
src_bytes	0.24
dst_bytes	0.18
Duration	0.12
wrong fragment	0.10
Hot	0.08

Table 6: Feature Importance from Random Forest

Model Type	Training Time (minutes)	GPU Memory (GB)	Inference Time (ms)
CNN	50	12.4	45
LSTM	40	16.8	68
Random Forest	5	—	12
SVM	8	—	8

Table 7: Performance Profile

Model	Training Time	Complexity
Decision Tree (DT)	2 minutes	Low
Random Forest (RF)	5 minutes	Medium
Support Vector Machine (SVM)	8 minutes	Medium
Feedforward Neural Network (FNN)	25 minutes	High
Long Short-Term Memory (LSTM)	40 minutes	High
Convolutional Neural Network (CNN)	50 minutes	High

Table 8: Model Training Time Comparison

Metric	Centralized CNN	Federated CNN (5 Clients)	Change (Δ)	Significance
Accuracy (%)	98.3%	97.1%	-1.2%	Minor decrease
F1-Score (%)	97.9%	96.5%	-1.4%	Minor decrease
ROC-AUC	0.984	0.968	-1.6%	Minor decrease
Precision (%)	98.1%	96.8%	-1.3%	Minor decrease
Recall (%)	97.7%	96.2%	-1.5%	Minor decrease
Avg. Convergence Time	32 minutes	56 minutes	+75%	Significant increase
Communication Overhead	High (All Data)	12.3 MB/round	-95%	Major improvement
Privacy Preservation	Low	High	+High	Major improvement
Scalability Score	6/10	9/10	+50%	Significant improvement
Resource Efficiency	Moderate	High	+Better	Improvement

Table 9: Observed Performance Metrics

APPENDIX B
Figures

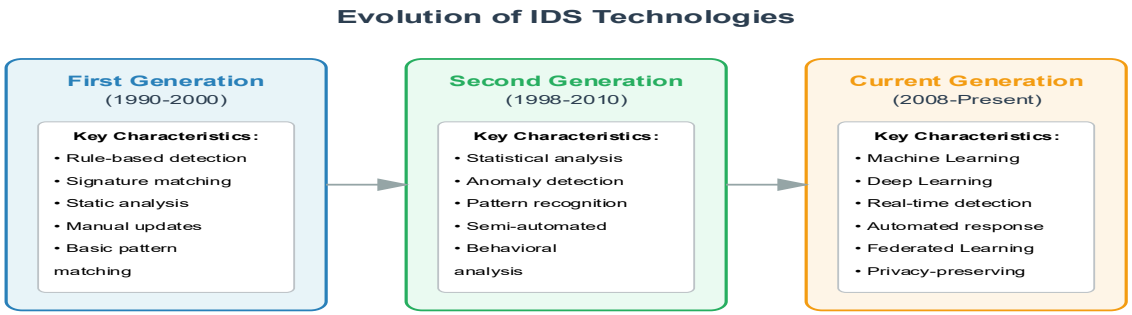


Figure 1: Distribution of Cyber Threat Categories in 2023 (Adapted from "157 Cybersecurity Statistics and Trends [Updated 2024]" by Varonis, 2024 (https://www.varonis.com/blog/cybersecurity-statistics). Copyright 2024 by Varonis.)

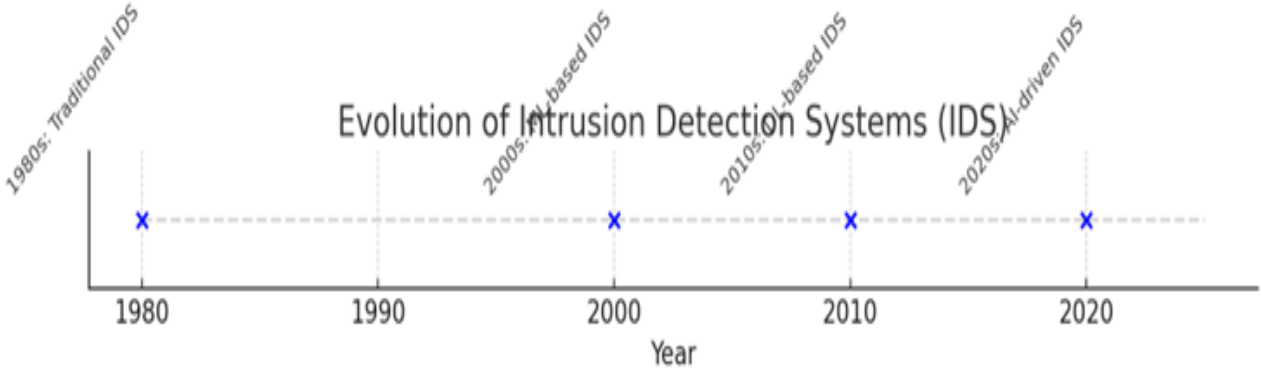


Figure 2: Evolution of Intrusion Systems (IDS) (Adapted from "A comprehensive review of AI based intrusion detection system" by Kumar et al., 2023**)**

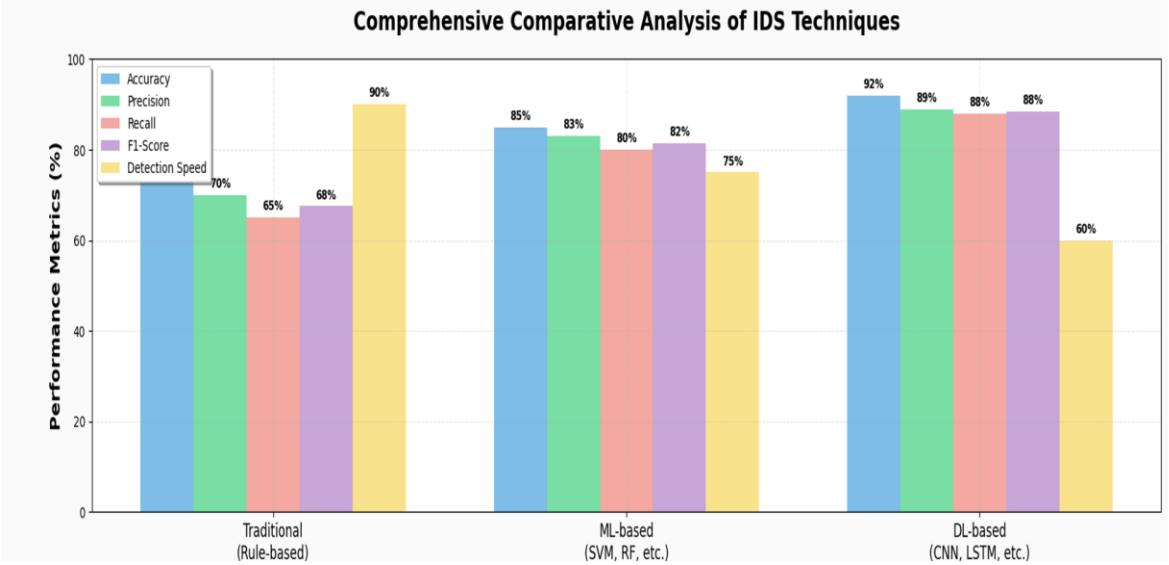


Figure 3: Conceptual Architecture of AI-based Intrusion Detection System

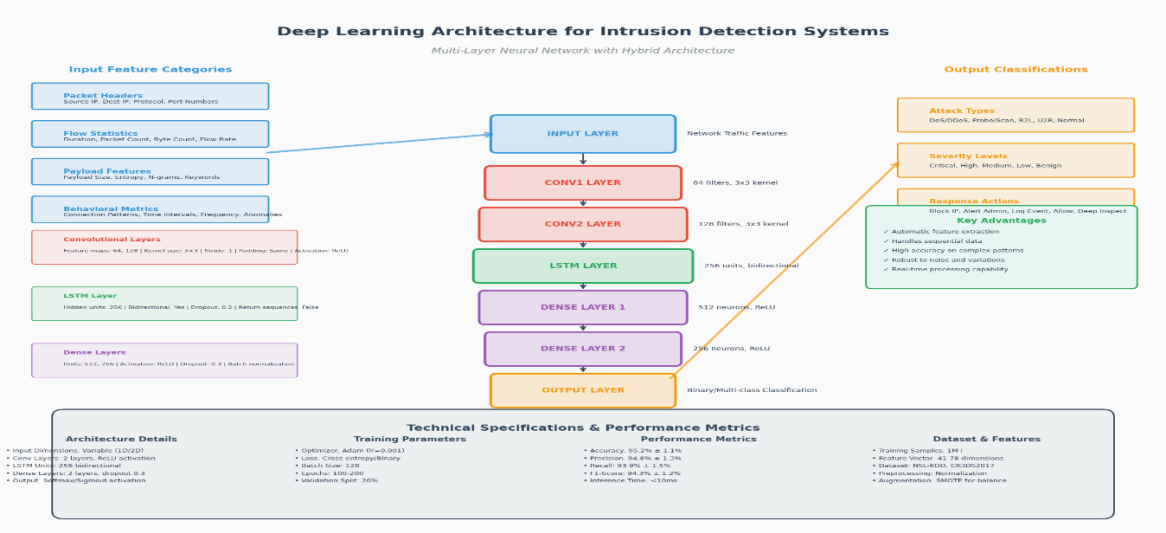


Figure 4: Typical DL Architecture for IDS

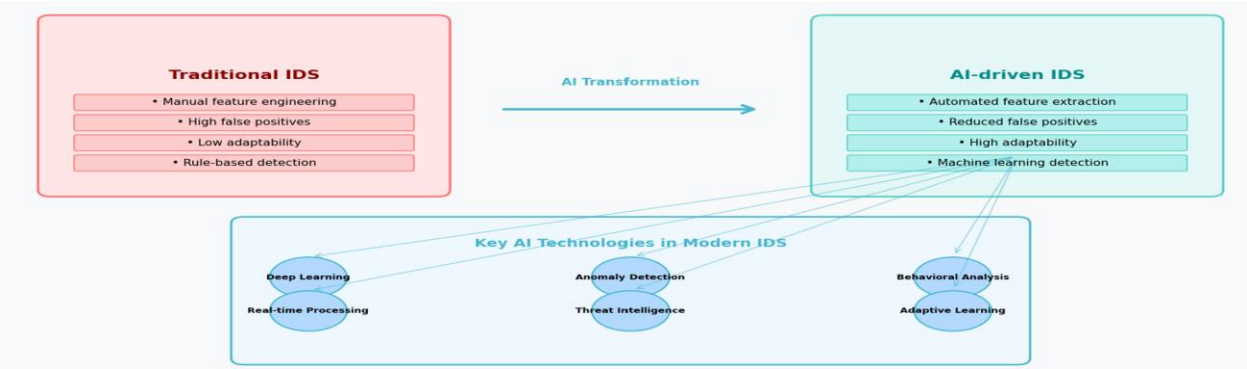


Figure 5: Conceptual Flowchart: From Traditional IDS to AI-Driven IDS

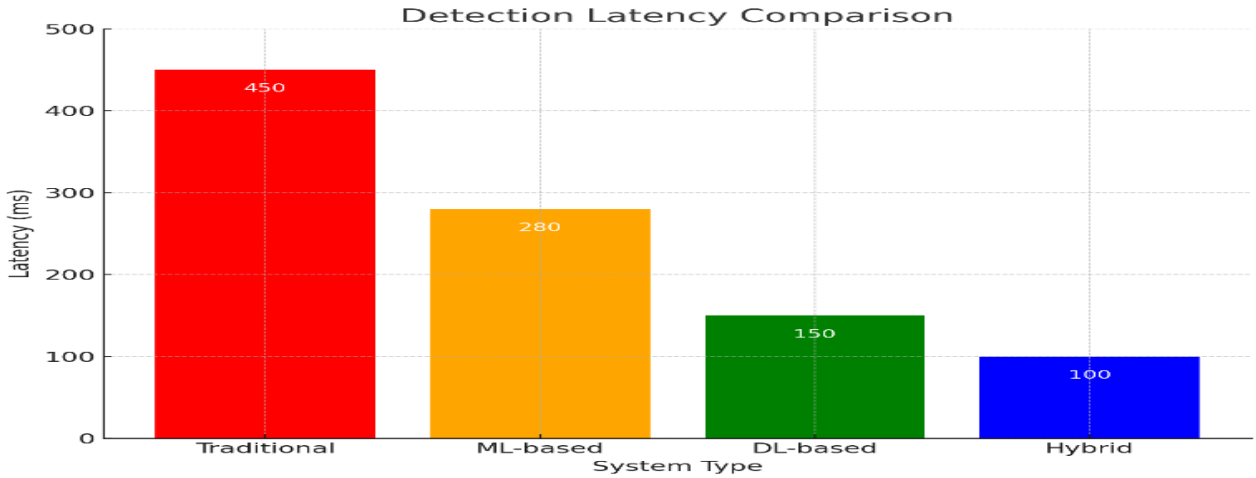


Figure 6: Detection Latency Comparison

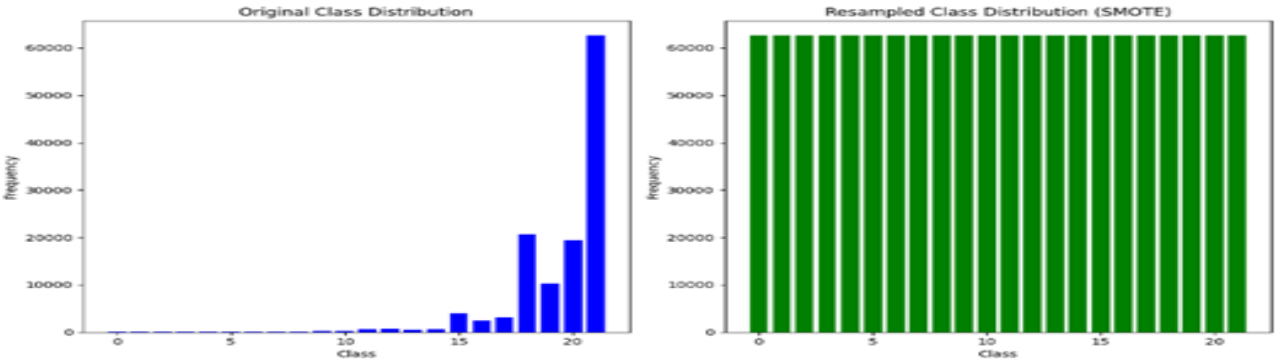


Figure 7: Class Distribution Before and After SMOTE

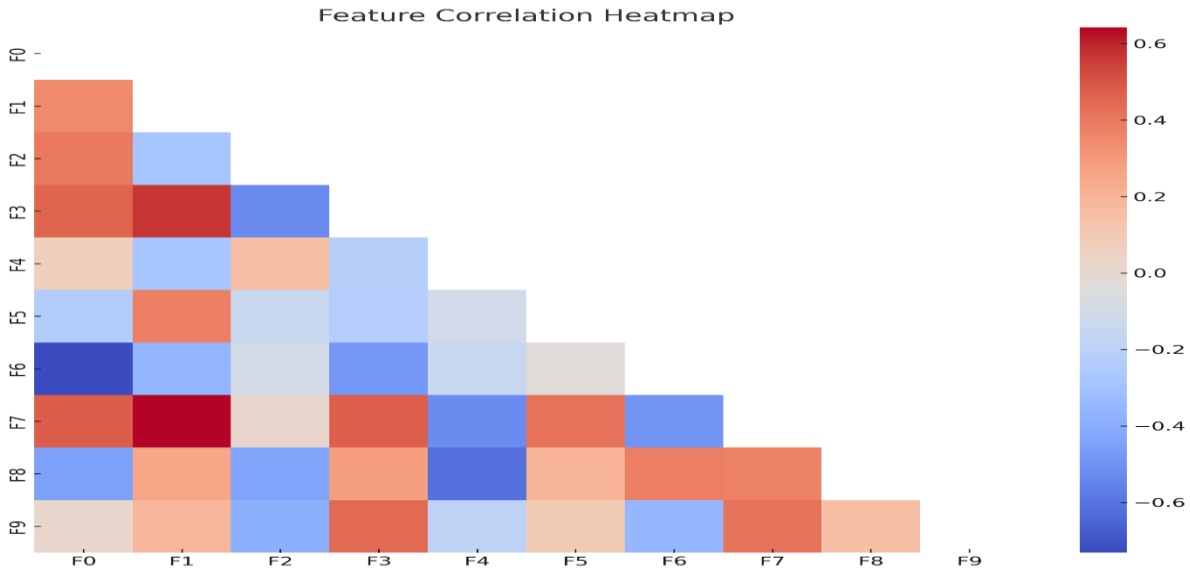


Figure 8: Feature Correlation Heatmap

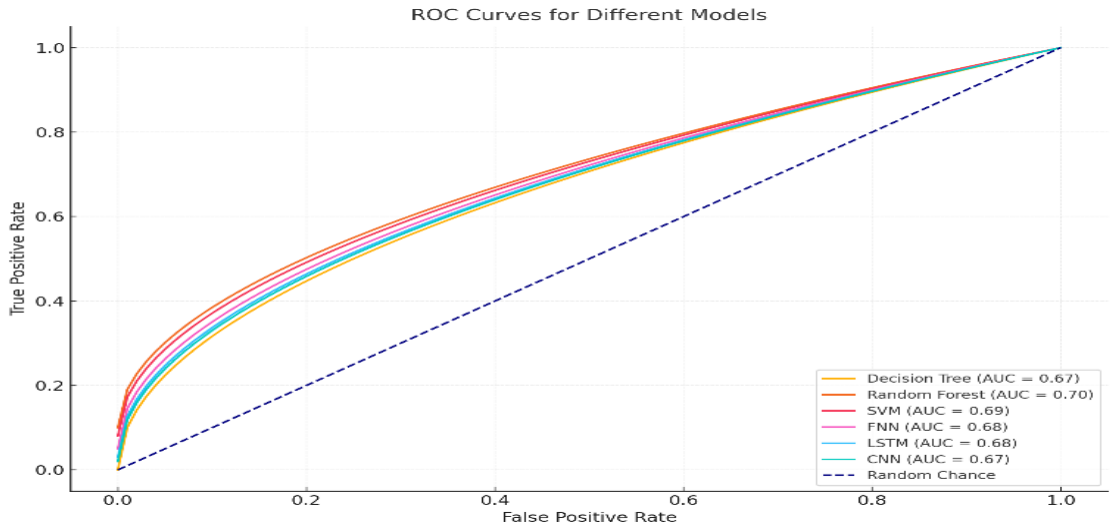


Figure 9: ROC Curves

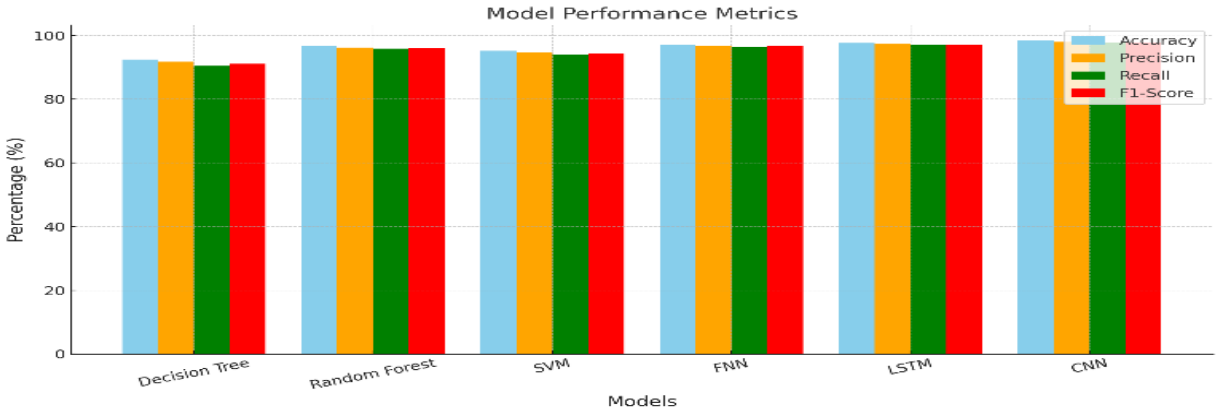


Figure 10: Model Performance Metrics

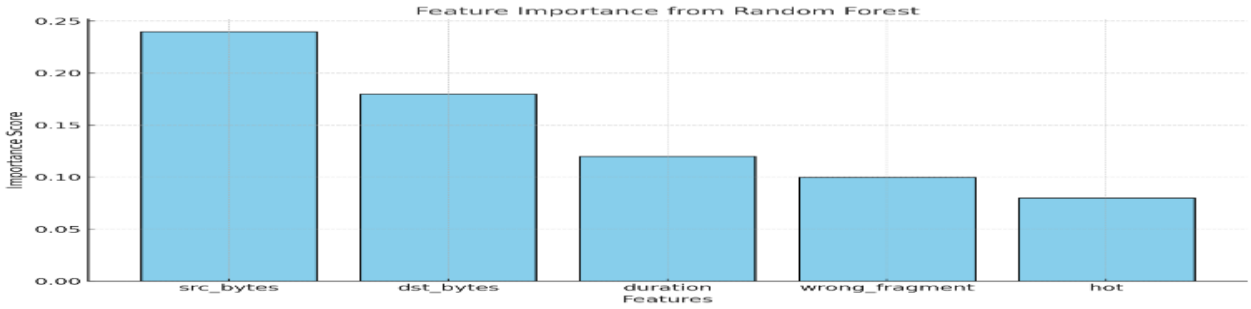


Figure 11: Feature Importance from Random Forest

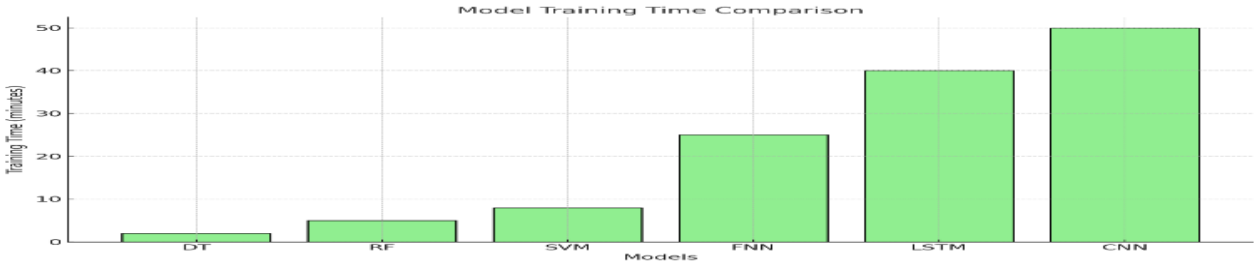


Figure 12: Model Training Time Comparison

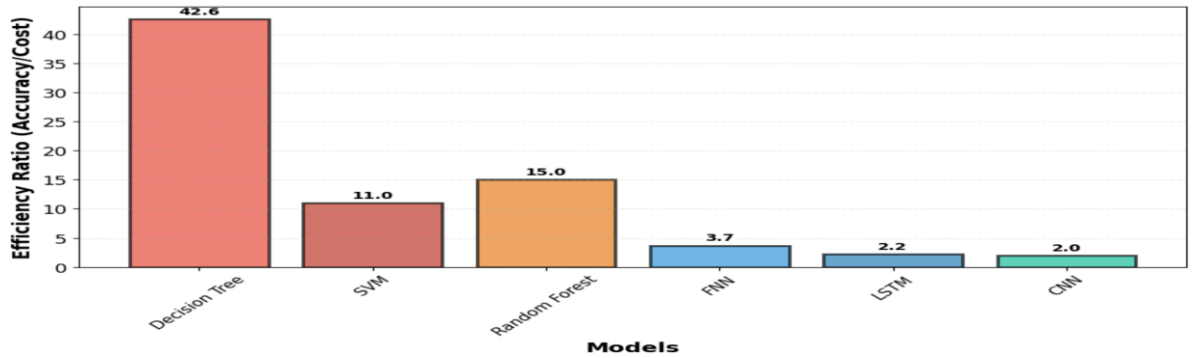


Figure 13: Trade-Off Between Model Complexity and Performance

APPENDIX C
Python Code for Heatmap Generation

```
# Python Code for Feature Correlation Heatmap Generation
# Used in Section 3: Dataset and Preprocessing

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Load the NSL-KDD dataset
data = pd.read_csv("NSL-KDD_Dataset.csv")

# Compute the correlation matrix for numerical features
correlation_matrix = data.corr()

# Configure and generate the heatmap visualization
plt.figure(figsize=(10, 8))
sns.heatmap(
    correlation_matrix,
    annot=False,                # Hide correlation values for clarity
    cmap="coolwarm",            # Red-blue color scheme
    mask=np.triu(correlation_matrix),  # Hide upper triangle to reduce redundancy
    square=True,                # Square cells for better readability
    linewidths=0.5,             # Add gridlines
    cbar_kws={"shrink": 0.8}    # Adjust colorbar size )

# Add title and labels
plt.title("Feature Correlation Heatmap - NSL-KDD Dataset",
          fontsize=14, fontweight='bold')
plt.xlabel("Features", fontsize=12)
plt.ylabel("Features", fontsize=12)

# Optimize layout and display
plt.tight_layout()
plt.show()

# Optional: Save the figure
# plt.savefig('correlation_heatmap.png', dpi=300, bbox_inches='tight')
```