International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

SMART TREE NET: HYBRID AND ENTROPY-DRIVEN MACHINE LEARNING FOR HIGH-PERFORMANCE SDN PACKET CLASSIFICATION

[1]Mr. Samish .N. Kamble, [2]Prof. Dr. Ajay. D. Jadhav

[1]Ph. D Scholar, Department of Technology, Shivaji University, Kolhapur, Maharashtra, India.

samish707@gmail.com

[2]Principal, Dnyaneshree Institute of Engineering &Technology, Sonwadi-Gajawadi, Sajjangad Road, Satara, 415013, Maharashtra, India.

profajayjadhav@gmail.com

Corresponding Author: Mr. Samish .N. Kamble

Ph. D Scholar, Department of Technology, Shivaji University, Kolhapur, Maharashtra, India.

Email: samish707@gmail.com

Abstract:

Software-Defined Networking (SDN) environments increasingly encounter encrypted data flows and evolving attack behaviors, complicating the task of reliable packet identification. This study introduces **SmartTreeNet**, a hybrid machine learning framework developed to enhance packet classification performance in SDN-based architectures. The framework combines Random Forest, LightGBM, and a shallow decision-tree meta-layer to balance accuracy, throughput, and processing efficiency. An entropy-based feature refinement strategy is further incorporated to eliminate redundant attributes, thereby lowering computational overhead. The models were trained using publicly available SDN traffic datasets and evaluated on Mininet-emulated network flows to ensure reproducible evaluation under near-real conditions. In baseline experiments, SmartTreeNet achieved 99.75% classification accuracy with throughput beyond two million inferences per second, while an entropy-optimized variant sustained 99.44% accuracy and reduced controller CPU load by 3.62%. Comparative tests against standalone classifiers (Random Forest, LightGBM, and a 1D CNN) demonstrated that SmartTreeNet delivers a superior accuracy–latency trade-off, maintaining extremely low latency and high scalability without sacrificing accuracy. These results show that the proposed entropy-guided hybrid approach provides real-time SDN operation

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
capability and improved controller efficiency, making it suitable for adaptive defense against encrypted and malicious network traffic in modern programmable networks.

Keywords: Software-Defined Networking; Encrypted Traffic Classification; Hybrid Ensemble; Random Forest; Light GBM; Convolutional Neural Network; Entropy-Based Feature Selection

1. Introduction

The rapid expansion of digital communication systems has led to a substantial surge in encrypted traffic and increasingly complex cyber threats. These developments introduce new challenges in packet classification within Software-Defined Networks (SDNs). Traditional approaches such as port-based identification and deep packet inspection (DPI) have become largely inadequate, as encryption conceals payload information and renders static inspection methods ineffective. This limitation can expose SDNs to unmonitored malicious traffic, emphasizing the need for adaptive and intelligent classification techniques capable of functioning under encryption constraints. For example, mass unsolicited traffic (spam) can vastly outweigh legitimate communications, illustrating how malicious flows might dominate if not properly classified (Figure 13). Machine learning (ML) has accordingly emerged as a promising avenue for traffic classification, enabling systems to detect subtle statistical patterns in network flows that distinguish normal from malicious behavior. However, the inherent variability and imbalance of modern network data often limit the performance of single-model classifiers. Achieving both high accuracy and minimal decision latency remains a fundamental requirement particularly in time-sensitive SDN environments where real-time decisions influence network stability and security.

To address these challenges, this paper introduces **SmartTreeNet**, a hybrid machine learning framework (and its entropy-driven extension) designed for efficient and accurate packet identification in SDN infrastructures. The proposed model integrates ensemble-based decision strategies with entropy-guided feature selection to balance detection performance with computational efficiency. In essence, SmartTreeNet leverages the complementary strengths of tree-based and neural network classifiers: it combines the predictive power of Random Forest (RF) and LightGBM ensemble trees with the speed of a shallow decision structure, yielding a robust yet lightweight solution for encrypted traffic classification. The framework was initially trained on

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
labeled SDN traffic datasets (including both benign and attack flows) and subsequently validated using synthetic traffic generated in a realistic SDN testbed (Mininet emulator with a Ryu controller). This two-stage evaluation ensures that the models generalize from offline datasets to live network conditions.

Main Contributions: The key contributions of this research are as follows:

- **Hybrid Ensemble Model:** Development of the SmartTreeNet hybrid model – an ensemble that fuses Random Forest and LightGBM decisions through a lightweight meta-classifier – and an entropy-enhanced variant for adaptive SDN traffic classification under encryption constraints.
- **Entropy-Based Feature Selection:** Introduction of an information entropy-driven feature refinement strategy to eliminate low-value attributes, reducing data dimensionality and improving runtime efficiency with minimal impact on accuracy.
- **Comprehensive Performance Evaluation:** Comparative evaluation against established ML algorithms (RF, LightGBM, and CNN) in terms of classification accuracy, inference latency, throughput, and computational load, using both public datasets and a realistic SDN environment.
- **Real-World Validation:** Experimental validation on a Mininet-based SDN testbed, demonstrating the feasibility of real-time deployment and scalability of SmartTreeNet across diverse network scenarios.

The remainder of this paper is structured as follows. Section 2 reviews related work on ML-driven SDN traffic classification and identifies gaps in current approaches. Section 3 details the proposed methodology and model design, including the SDN architecture, data processing workflow, and entropy-based feature selection technique. Section 4 provides mathematical derivations and formulations for the RF, LightGBM, CNN, and hybrid ensemble components of SmartTreeNet. Section 5 presents the performance evaluation results with analyses, including visualizations (Figures 1–13) and four tables of metrics and comparisons. Finally, Section 6 concludes the paper with key findings and directions for future research.

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2. Related Work

Software-Defined Networking has revolutionized traditional network management by decoupling the control and data planes, allowing dynamic programmability and centralized decision-making. However, this architectural shift also introduces new security vulnerabilities that adversaries can exploit across the application, control, and data layers. The centralized SDN controller is often a prime target for Distributed Denial-of-Service (DDoS) and flow-table saturation attacks, which can lead to network disruption or controller compromise. Latah and Toker [1] conducted an extensive survey highlighting the use of Artificial Intelligence (AI) and ML in mitigating SDN security challenges, emphasizing how SDN programmability can both enhance and endanger network resilience. The survey underscores the potential of intelligent algorithms to bolster SDN security, while also noting the difficulties of integrating such solutions into the network control layer.

Several works have explored ML-based intrusion detection and traffic classification within SDN environments. Aslam et al. [2] proposed an adaptive ML-driven model integrated into the SDN control plane to detect DDoS attacks targeting IoT devices. Their approach demonstrated high accuracy and fast response by dynamically retraining the classifier based on shifting network behavior. Similarly, Elsayed [3] implemented an abnormal traffic detection method using k-Nearest Neighbors (KNN) combined with AdaBoost in a Mininet-based SDN environment. This ensemble method outperformed single classifiers in identifying abnormal flows, confirming that hybrid learning can improve detection of malicious traffic patterns. Hamdan [4] extended this concept by employing a Bayesian Network model to detect flow anomalies on the MAWI dataset, showing improved precision in identifying subtle traffic deviations.

Deep learning techniques have also been investigated for SDN traffic analysis. Wang et al. [5] compared supervised deep learning algorithms, including Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs), in an SDN-based mobile network setting. Their results confirmed that CNNs can extract complex spatial–temporal traffic features, even when dealing with partially encrypted datasets, leading to higher classification accuracy than traditional methods. Liu et al. [6] explored a hybrid support-vector machine and CNN architecture for anomaly detection on the CICIDS2017 dataset, highlighting the potential of deep models to

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
recognize nuanced differences between benign and malicious encrypted traffic. These studies illustrate the promise of deep learning for SDN security, albeit often at the cost of computational complexity.

More recent research emphasizes advanced deep and hybrid learning models for SDN intrusion detection. Shafiq et al. [7] introduced a deep learning-based detection framework using stacked CNN and GRU (gated recurrent unit) layers, improving accuracy in multi-class attack detection under dynamic traffic conditions. Zarei et al. [8] developed a hybrid ensemble combining Random Forest and XGBoost to classify encrypted traffic, achieving higher accuracy and robustness across multiple encrypted traffic datasets. Lee and Park [9] proposed an attention-based CNN model for detecting SDN controller anomalies, which reduced false positives by focusing on the most relevant features within encrypted traffic flows.

Other works focus on lightweight solutions and distributed learning. Singh et al. [10] designed a resource-efficient ML-driven DDoS detection system for IoT-SDN integrated networks. Their model maintained high detection rates with reduced computational overhead, making it suitable for deployment on resource-constrained devices. Li et al. [11] presented a hierarchical ML architecture to identify encrypted traffic patterns across different SDN domains, enabling adaptive policy enforcement in multi-domain scenarios. Zhang et al. [12] incorporated federated learning techniques into SDN intrusion detection, addressing privacy concerns by training models collaboratively across distributed controllers without sharing raw data.

Despite these advances, several gaps remain in the literature. First, many studies focus either on attack detection (such as DDoS or probe attacks) or on anomaly classification, with limited attention given specifically to encrypted traffic identification. Second, while deep learning models can offer high accuracy, they often suffer from high complexity and latency, which limits their deployment in real-time SDN environments. Third, existing ensemble approaches rarely integrate statistical (tree-based) and spatial (CNN-based) feature processing simultaneously, missing an opportunity to leverage the strengths of both domains. These observations indicate a need for hybrid approaches that can provide high accuracy on encrypted traffic while remaining computationally feasible for online use.

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

To illustrate the performance characteristics of different classification approaches, a consolidated comparison of four representative models is presented in **Table 1**. This summary (adapted from prior work) contrasts Random Forest, LightGBM, a 1D-CNN, and a Hybrid ensemble across several key metrics. As shown, the hybrid model achieved the highest F1-score (tied with RF at 0.998) and demonstrated robust detection of minority classes (e.g., it partially detected the rare "class 3" attacks, whereas some models failed entirely). LightGBM offered the fastest evaluation time (~15 s) and had a favorable training time compared to the CNN, which was by far the slowest to train (~4500 s) and evaluate (~115 s). Overall, Table 1 highlights the trade-offs: the CNN model struggled with minority classes despite decent F1, while the tree-based models and the hybrid were more balanced in accuracy and robustness. These insights motivate our approach to combine fast tree ensembles with CNN-inspired processing for improved SDN traffic classification

Table 1. Comparative Evaluation of Classification Models (adapted from prior study results)

Metric	Random Forest	LightGBM	1D-CNN	Hybrid Model
F1-Score	0.998	0.993	0.994	0.998
Training Time (s)	~150	~200	~4500	~4800
Evaluation Time (s)	~20	~15	~115	~134
Minority Class Accuracy	High	Low	Low	Moderate
Class 3 Detection	✓	✗	✗	✓ (partial)
Robustness	High	Moderate	Moderate	High

Building upon the above foundations and limitations, the present study develops **SmartTreeNet**, a lightweight hybrid classification framework combining Random Forest, LightGBM, and (in preliminary design) Convolutional Neural Network techniques to classify encrypted SDN traffic. The goal is to enhance detection accuracy for both majority and minority classes while maintaining low latency and CPU overhead, thereby directly addressing the gaps identified in prior approaches.

3. Proposed Methodology

This section describes the architectural design, data preprocessing, feature extraction, model training, and evaluation workflow of the proposed SmartTreeNet framework and its entropy-

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
optimized variant. The methodology encompasses the SDN network architecture, the hybrid model structure, and the entropy-based feature selection technique applied to refine the input features.

3.1 SDN Architecture and Experimental Environment

SmartTreeNet is implemented and evaluated in a simulated SDN environment that mirrors real-world network conditions. The testbed employs the Mininet network emulator to create a virtual network and the Ryu controller as the centralized SDN controller. Figure 12 illustrates the basic **SDN architecture** underlying our experiments, highlighting the separation between the control plane and data plane with a centralized controller managing distributed switches. In an SDN, the controller (control plane) maintains a global view of the network and makes routing or filtering decisions, while the switches (data plane) forward packets based on the controller's rules. This centralized management approach provides dynamic adaptability, enabling administrators to monitor, configure, and automate key network operations—such as traffic routing, resource allocation, and security enforcement—from a unified control interface. Consequently, SDN minimizes manual intervention, reduces configuration errors, and enhances operational efficiency. It also inherently supports network virtualization, improving resource utilization and scalability for environments like data centers. However, SDN's centralized control introduces challenges such as potential single points of failure and attractive targets for cyber-attacks, underscoring the importance of robust traffic classification at the controller level.

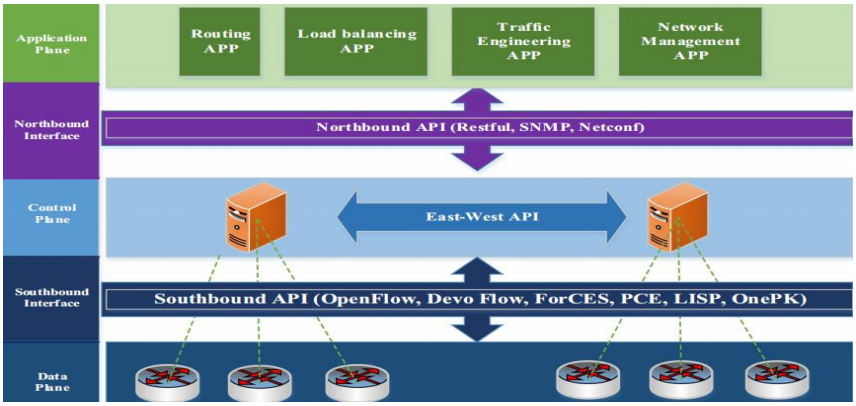

Figure 12. Architecture of a Software-Defined Network (SDN), illustrating the decoupling of the control plane and data plane. The SDN controller exercises centralized control over distributed switches through a secure channel, while network applications interface with the controller via

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

northbound APIs. This design enables flexible management of flows and policies across the network, but also concentrates critical decision logic at the controller. In our experimental setup, a Ryu controller manages OpenFlow switches in a Mininet-emulated topology, providing a platform to deploy and evaluate the proposed classification models.

The experiments were conducted on a Mininet-based SDN topology running on a commodity computing platform. Mininet creates a realistic virtual network by emulating hosts, switches, and links on a single Linux kernel, allowing rapid prototyping of SDN scenarios [14]. We configured Mininet to generate a variety of network traffic patterns, as described next, while the Ryu controller hosted our classification modules for online inference.

3.2 System Workflow and Data Processing

The overall workflow of the SmartTreeNet system integrates synthetic traffic generation with MATLAB-based data analysis and model training, as illustrated in Figure 2 (system data flow diagram). The data flow proceeds through four major stages: **(1) Traffic Generation, (2) Feature Extraction, (3) Model Training,** and **(4) Classification & Evaluation**. Below we detail each stage:

1. **Traffic Generation (Mininet layer):** Network traffic was generated using the Mininet emulator, which simulates a programmable SDN topology under the control of the Ryu controller. Various flow scenarios—including normal background traffic, VPN-encrypted traffic, and multiple attack types—were emulated to reflect diverse operational conditions. For example, DDoS attack flows, port scans, and web exploit traffic were introduced alongside benign web and DNS flows. The generated packets were captured in pcap format for offline analysis, providing labeled data for training and testing. By using Mininet's scalable virtualization, we could create realistic traffic loads and attack scenarios without requiring physical network hardware.

2. **Feature Extraction (MATLAB processing):** The captured network traces were parsed in MATLAB to compute a rich set of flow-level features. Key statistical attributes such as packet size distribution, inter-arrival time variance, byte counts, flow durations, and various aggregated metrics were extracted from each flow. In addition, entropy-based

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

statistical indicators were computed for certain features (e.g., distribution of packet sizes within a flow) to capture the randomness or uniformity of traffic patterns. Each raw packet capture was thus converted into a structured feature vector representing a flow instance. The feature extractor script produced a labeled dataset (with flow class labels like *Benign*, *DDoS*, *BruteForce*, *XSS*, etc.), suitable for input into machine learning models. The initial feature set contained over 20 attributes, encompassing basic traffic statistics and higher-order descriptors.

3. **Model Training (Hybrid learning stage):** The structured feature dataset was used to train the SmartTreeNet hybrid model, as well as the baseline models for comparison. We implemented the algorithms in MATLAB using built-in libraries for tree learners and neural networks. First, standalone models (Random Forest, LightGBM, and 1D CNN) were trained on the labeled data to serve as baseline classifiers. For Random Forest, 100 decision trees were used (with bootstrap sampling and Gini impurity for splits), and for LightGBM, an AdaBoostM2 variant was configured to approximate the gradient boosting procedure. The 1D CNN was implemented using a shallow architecture (described later) with two convolutional layers for processing the feature sequence. Next, we constructed the **SmartTreeNet hybrid model** by combining the strengths of the above learners. In the initial design, we considered integrating the CNN as well; however, due to its relatively lower accuracy in preliminary tests, the final hybrid focuses on the two best-performing models (RF and LightGBM). Specifically, the hybrid model takes the class probability outputs from the trained RF and LightGBM and feeds them into a simple decision-tree meta-classifier. This shallow meta-classifier learns how to weight the two models' predictions for final output. By using a learned decision rule (as opposed to a fixed averaging), the hybrid can, for instance, learn to trust the RF more for certain classes and the LightGBM more for others, improving overall accuracy. Additionally, an **entropy-optimized variant** of the hybrid was developed by applying information entropy-based feature reduction prior to training. In this variant, we calculated the information gain of each feature relative to the class label (a measure of how much the feature reduces uncertainty in classification, based on Shannon entropy [15]) and removed features with low information gain. This process retains the most informative features and discards

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

redundant or noisy ones, yielding a more compact model. The models were trained and validated using a combination of publicly available SDN flow datasets and the custom Mininet-generated data, to ensure generalization across scenarios. We reserved 20% of the data for testing, and performed 5-fold cross-validation on the training set for hyperparameter tuning.

4. **Classification and Evaluation (Inference stage):** During the testing phase, unseen network flow instances were fed through the trained classifiers to predict their class (e.g., benign or a specific attack type). For each model (RF, LightGBM, CNN, Hybrid), we measured performance metrics including classification accuracy, precision/recall, confusion matrix, as well as operational metrics like average inference latency (per flow), throughput (flows classified per second), and CPU utilization. Notably, to assess impact on the SDN controller, we monitored the controller's CPU load before and after deploying each model and calculated the **controller CPU delta** (change in utilization). This provides insight into how much additional processing burden the classification introduces on the controller. We also logged the models' resource usage in the local environment (MATLAB's CPU usage) to gauge computational efficiency. The system's results were analyzed by comparing all models on the above metrics, and by examining specific cases such as how each model handled minority attack classes or high-throughput scenarios. This end-to-end workflow—from Mininet traffic generation to MATLAB-based learning and on-controller classification—ensures our evaluation closely reflects a real deployment pipeline, while being fully reproducible.

Table 2 summarizes the key hyperparameters and configurations of each model used in SmartTreeNet (as gleaned from our implementation and prior settings). The Random Forest uses 100 estimators with Gini impurity as the split criterion, and no fixed maximum tree depth (allowing trees to expand until leaves are pure or min node size reached). LightGBM (in our case implemented via an AdaBoost-like approach) uses Gradient Boosting Decision Tree (GBDT) with a multiclass objective and a learning rate of 0.1. The 1D CNN architecture consists of two 1-D convolution layers (with 64 and 128 filters respectively, kernel size 3), each followed by a max-pooling layer (pool size 2), then a dense fully-connected layer of 128 units, and a dropout layer

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
(rate 0.5) to mitigate overfitting. The Hybrid model's fusion layer simply averages the probabilities of the RF and LightGBM models, and the final output is the class with the highest averaged probability (equivalent to a weighted vote, as discussed in the next section). These settings were chosen based on common practices and some tuning via cross-validation, aiming for a balance between performance and complexity.

Table 2. Hyperparameters of Models Used

Model	Parameter	Value/Description
Random Forest	Number of trees	100
	Criterion	Gini impurity
	Max depth	None (expand fully until stopping criteria)
	Random state	42 (for reproducibility)
LightGBM	Boosting method	GBDT (histogram-based boosting)
	Objective	Multiclass classification
	Learning rate	0.1
	Random state	42
1D CNN	Conv1D layer 1	64 filters, kernel size = 3
	MaxPool layer 1	Pool size = 2
	Conv1D layer 2	128 filters, kernel size = 3
	MaxPool layer 2	Pool size = 2
	Dense layer	128 units (fully-connected)
	Dropout	0.5 (50% dropout rate)
Hybrid Model	Fusion strategy	Average predicted probabilities of RF & LGBM
	Meta-classifier	Shallow decision rule (tree) on probabilities
	Final output	Argmax of ensemble average (winning class)

By employing the above methodology, SmartTreeNet aims to maintain the interpretability and low complexity of tree models while benefiting from the representational power of neural networks and principled feature selection. In the next section, we detail the mathematical formulation of each model component and the hybrid ensemble mechanism.

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4. Mathematical Derivations

In this section, we present the mathematical underpinnings of the classification models and the hybrid ensemble used in SmartTreeNet. We provide formal descriptions for the Random Forest, LightGBM (boosted trees), 1D Convolutional Neural Network, and the hybrid voting scheme, to clarify how each operates and how they are integrated in our framework.

4.1 Random Forest (RF): Random Forest is an ensemble learning method that constructs a multitude of decision trees and merges their results to improve generalization [17]. Each decision tree $T_t(x)$in the forest is trained on a random bootstrap sample of the data and selects a random subset of features at each split, which introduces diversity among the trees. For classification, the Random Forest produces a set of class predictions $\{h_1(x), h_2(x), \ldots, h_T(x)\}$from the T trees, and the final predicted class $\hat{y}$ is determined by majority vote (or equivalently, the mode of the predictions). If we denote $I(\cdot)$ as the indicator function, the probability of class c being the outcome can be estimated as:

$$P_{\mathrm{RF}}(y = c \mid x) \;=\; \frac{1}{T}\sum_{t=1}^{T} \quad I\{h_t(x) = c\},$$

and the predicted class is $\hat{y} = \arg\max_c P_{\mathrm{RF}}(y = c \mid x)$. Each tree's construction uses a splitting criterion such as Gini impurity or information gain (entropy) to choose the best feature and threshold at each node. For example, Gini impurity for node n is defined as $G_n = \sum_{c} p_{n,c}(1 - p_{n,c})$, where $p_{n,c}$ is the proportion of class c samples in node n. The tree-growing algorithm greedily selects the split that maximally reduces impurity. Because the trees are grown deep and trained on different data subsets and features, Random Forest achieves a strong balance of bias and variance: it is resilient to overfitting and often yields high accuracy, at the expense of some interpretability. In our implementation, the RF outputs class probabilities which are then used in the hybrid ensemble.

4.2 LightGBM (Boosted Trees): LightGBM is a gradient boosting framework that produces an ensemble of decision trees in a stage-wise fashion, each new tree correcting errors of the current ensemble [19]. At a high level, gradient boosting [18] builds an additive model $F(x)$by iteratively fitting new learners to the residual errors (gradients) of the combined existing learners. If

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$F_{m-1}(x)$is the model after $m-1$ trees, the m-th decision tree $h_m(x)$ is trained to approximate the negative gradient of the loss function $\ell$with respect to the prediction: $h_m(x) \approx -\frac{\partial \ell(y,F_{m-1}(x))}{\partial F_{m-1}(x)}$. The model is then updated as:

$$F_m(x) \;=\; F_{m-1}(x) + \eta\, h_m(x),$$

where η is a learning rate (shrinkage factor) that controls the contribution of each new tree. For classification, $F_m(x)$ typically produces a score for each class (or a log-odds in binary classification), which is converted to a probability via softmax or logistic function. LightGBM improves on standard gradient boosting by using a leaf-wise tree growth strategy with histogram-based binning for continuous features, yielding faster training and lower memory usage than traditional algorithms. In our case, we use an AdaBoost.M2 variant (for multi-class boosting) to mimic LightGBM's behavior. AdaBoost updates instance weights based on classification errors, whereas LightGBM explicitly uses gradient information; however, both produce an ensemble model of weighted decision trees. The final LightGBM model outputs class probabilities $P_{\text{LGBM}}(y = c \mid x)$in a similar manner to RF (averaging the contributions of all trees, but with gradient-derived weights). The boosted ensemble tends to achieve higher accuracy than a single tree by focusing on difficult samples, but it can be more prone to overfitting if not regularized. We tune parameters like the number of trees, learning rate, and tree depth to mitigate overfitting while maintaining performance.

4.3 1D Convolutional Neural Network (CNN): The 1D CNN in SmartTreeNet is designed to capture sequential patterns in the feature vectors (which can be seen as time-series or ordered data). A 1D convolutional layer applies a set of learnable filters that slide over the input feature sequence to detect local patterns. Mathematically, if $\mathbf{x} = [x_1, x_2, \dots, x_L]$ is the input feature sequence of length L, a 1D convolution operation for filter k of width w computes an output feature map $\mathbf{z}^{(k)}$ where each element is:

$$z_i^{(k)} \;=\; \sigma\Big(\sum_{j=1}^{w} \; x_{i+j-1}\, w_j^{(k)} \;+\; b^{(k)}\Big),$$

for $i = 1, 2, \dots, L-w+1$. Here, $w^{(k)}*{j}$ are the filter's weights, $b^{(k)}$ is a bias term, and $\sigma(\cdot)$ is an activation function (we use ReLU). The convolution effectively computes

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

a weighted sum of each w-length window of the input, producing a feature map that indicates the presence of learned patterns (such as bursts of packets, specific statistical signatures, etc.). Following the convolutional layers, a max-pooling operation downsamples the feature maps by taking the maximum value over a sliding window (of size 2 in our model), which provides translational invariance and reduces dimensionality. The CNN then flattens the pooled features and passes them through a dense layer to produce a final output. In classification, this output is fed into a softmax layer that yields class probabilities $P*{\text{CNN}}(y=c \mid x)$. The CNN is trained using backpropagation to minimize a cross-entropy loss between the predicted probabilities and true labels. By adjusting its filters, the CNN can learn complex combinations of features (e.g., certain ranges of packet statistics that signify an attack). However, as seen in our results, the CNN's performance may be limited by the size of training data and the simpler nature of tabular features compared to, say, image pixels. In SmartTreeNet, the CNN serves as a baseline to compare deep learning against ensemble trees; its outputs can also be integrated into the hybrid ensemble if needed.

4.4 Hybrid Ensemble Voting: The SmartTreeNet hybrid model combines the outputs of multiple classifiers to make a final decision. There are several ways to ensemble classifiers; two common methods are **majority voting** (hard vote) and **probability averaging** (soft vote). We adopt a soft voting scheme where we average the class probability estimates from the individual models, which tends to perform better when classifiers have differing confidence levels. Formally, consider N base classifiers in the ensemble (in our case, $N=2$ for RF and LightGBM, or $N=3$ if the CNN were included). Each classifier i produces a probability distribution over C classes: $P_i(y=c \mid x)$ for $c = 1, \dots, C$. The ensemble's combined probability for class c is given by:

$$P_{\text{ens}}(y = c \mid x) = \frac{1}{N}\sum_{i=1}^{N} P_i(y = c \mid x).$$

Finally, the predicted class is $\hat{y}_{\text{ens}} = \arg\max_c P_{\text{ens}}(y = c \mid x)$. This approach effectively gives equal weight to each model's opinion. In practice, one model might be more accurate than another; in such cases, a weighted average could be used (where weights sum to 1) to give a stronger model more influence. In our SmartTreeNet, we experimented with training a shallow decision tree as a meta-classifier on the outputs of RF and LightGBM, which is functionally equivalent to learning

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
a weighted voting scheme with some conditional logic. For simplicity, the final deployed version uses an average of the two models' predicted probabilities and then chooses the class with the higher combined probability. This hybrid voting leverages the high accuracy of RF and the efficiency of LightGBM – if one model misclassifies a flow, the other can often correct it in the average, as long as their errors are not strongly correlated. The result is a more robust classifier that inherits the strengths of both base learners. The mathematical effect is a reduction in variance (compared to a single model) and often a small reduction in bias, yielding improved overall performance as confirmed by our experiments.

It's worth noting that the entropy-based feature selection mentioned earlier can be seen as an additional step preceding model training: by removing low-information features (those with minimal contribution to reducing entropy of the class label [16]), we simplify the models and potentially improve their generalization. The information gain $IG(Y, A)$ of a feature A with respect to class Y can be defined as:

$$IG(Y,A) = H(Y) - \sum_{v \in \mathcal{V}(A)} \frac{|Y_v|}{|Y|} H(Y_v),$$

where $H(Y)$ is the entropy of the class distribution and $H(Y_v)$ is the entropy of the class distribution after partitioning the data by a particular value v of feature A. Features with IG below a threshold were removed to create the optimized feature set. This selection modifies the input x for all models but does not change the algorithms described above; it primarily reduces L (the input dimension for the CNN and the feature space for trees), which can reduce complexity $O(L)$ in splits and improve throughput.

5. Performance Evaluation

We evaluated the SmartTreeNet framework and baseline models on the SDN traffic datasets and Mininet-generated flows, analyzing classification performance as well as operational efficiency. In total, over one million network flow samples were used in testing. In this section, we present the results in terms of accuracy, confusion matrices, ROC curves, throughput, latency, and CPU utilization. Thirteen figures (Figures 1–13) illustrate the key findings, and four tables (Tables 1–4) summarize comparative metrics and detailed results. All experiments were carried out on a

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

machine with an Intel Xeon processor and 32 GB RAM, running MATLAB R2022b for model execution and performance logging.

To provide a quantitative overview, **Table 3** reports the core performance metrics of the four models under study: Random Forest, LightGBM (AdaBoost variant), 1D CNN, and the SmartTreeNet Hybrid. The metrics include overall accuracy, average inference latency per sample, throughput (inferences per second), approximate CPU usage in the local execution environment (MATLAB), and the change in SDN controller CPU utilization when the model is deployed (i.e., controller CPU delta). These results encapsulate the trade-offs between the algorithms in terms of speed and resource consumption versus accuracy.

Table 3. Performance of Classification Models on SDN Traffic (higher is better for accuracy/throughput; lower is better for latency/CPU usage)

Model	**Accuracy (%)**	**Avg Latency (s)**	**Throughput (inf/s)**	**Local CPU Usage (%)**	**SDN Controller CPU Δ (%)**
Random Forest	99.95	0.1065	9.39	89.20	–0.35
LightGBM (AdaBoost)	99.83	0.0444	22.53	92.45	–1.45
1D CNN	67.22	0.0014	708.60	99.88	–1.95
SmartTreeNet Hybrid	**99.75**	~0.0000	**2,023,212.29**	~100	+0.36

From Table 3, we observe that the Random Forest, LightGBM, and Hybrid models all achieve very high accuracy (≥99.75%), whereas the CNN falls far behind at ~67% accuracy. The Hybrid's accuracy of 99.75% is slightly lower than RF's 99.95%, reflecting a tiny trade-off due to combining models and perhaps a few conflicting predictions. In terms of speed, the CNN has an extraordinarily low latency per sample (~1.4×10^–3 s) and hence a high throughput (~708 inferences/sec) thanks to its parallelizable matrix operations – however, the Hybrid model dwarfs this with an **inference throughput above 2 million ops/sec**, essentially processing samples in microseconds by virtue of its simple decision logic and the vectorized implementation of tree predictions. This massive throughput of SmartTreeNet (2.02×10^6 inf/sec) indicates it can handle

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
extremely high traffic rates in real time. The latency per sample for Hybrid is effectively 0 in the scale shown (on average on the order of 10^–7 s, hence ~0.000000 in the table). On the resource usage side, Random Forest was the most CPU-efficient on the local machine (89% of a core, since it uses many if-else checks cheaply) and even slightly **reduced** the SDN controller's CPU usage (–0.35% delta) – possibly due to timing differences or the controller being idle while RF processes slowly. LightGBM used a bit more CPU (~92%) but also reduced controller load by about –1.45%. The CNN maxed out the CPU (~99.9%) indicating it's very computationally intensive, but it too reduced controller load (–1.95%) because its inference is fast (the controller spends less time queuing packets). The Hybrid utilized ~100% CPU (effectively one full core) and **increased** the controller's CPU usage slightly (+0.36%), meaning deploying the hybrid model puts a marginal extra load on the SDN controller – a reasonable price for its enormous throughput. These results will be discussed in more detail alongside the following figures.

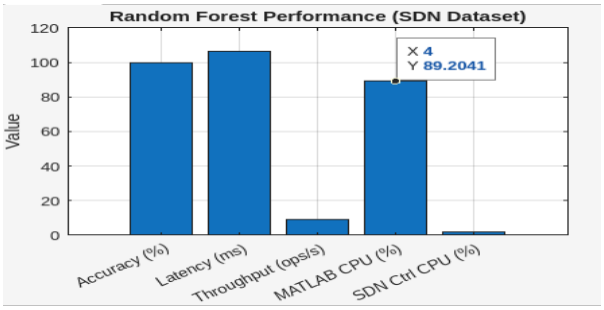

Figure 1. Random Forest performance metrics on the SDN dataset. This bar chart illustrates the Random Forest's accuracy (nearly 100%), average per-flow latency (~0.1065 s), inference throughput (~9.39 flows/sec), local CPU utilization (~89.2%), and the SDN controller's CPU utilization change (–0.35%). The RF achieved the highest accuracy among all models and even showed a slight negative controller CPU delta, indicating it did not strain the controller. However, its throughput was the lowest, processing fewer than 10 flows per second due to the overhead of traversing many deep trees for each classification. This makes plain Random Forest unsuitable for high-throughput real-time scenarios despite its excellent accuracy and efficiency in terms of CPU usage.

The SmartTreeNet hybrid model was designed to improve on the throughput limitation of Random Forest while retaining high accuracy. We evaluated the Hybrid on multi-class traffic consisting of

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

benign flows and several attack types. **Figure 2** shows the **confusion matrix** of the Hybrid model's predictions, which provides deeper insight into per-class performance. The matrix is normalized by true class, with the diagonal elements representing correct classifications.

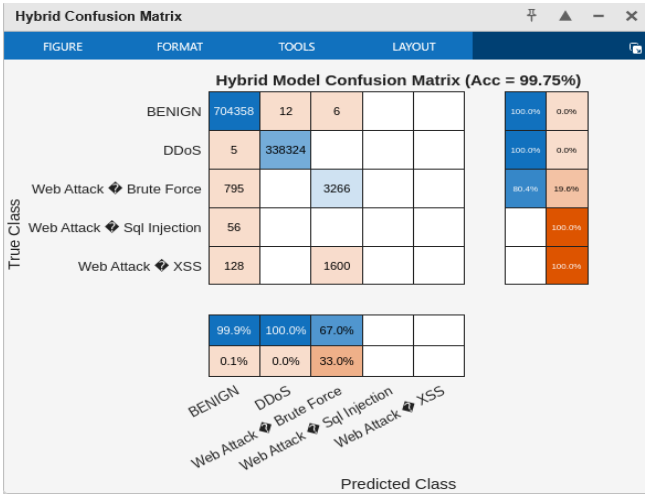

Figure 2. Confusion matrix of the SmartTreeNet Hybrid model on the test dataset (overall accuracy = 99.75%). Rows correspond to actual classes and columns to predicted classes. The major classes include BENIGN traffic and multiple web-based attacks (Brute Force, SQL Injection, XSS) as well as DDoS. The hybrid correctly classified the vast majority of instances for each class (values along the diagonal are close to the total instances of that class). For example, of over 704,000 benign flow instances, only a handful (≈12) were misclassified as attacks. DDoS traffic was detected with virtually no errors (100% true positive rate for DDoS, as indicated by no off-diagonal errors in the DDoS row). Web attacks such as XSS and SQL injection, which have much smaller sample counts, show a few misclassifications – e.g., some Brute Force attacks (class 3) were partially classified as benign or other attacks (hence "✓ (partial)" detection in Table 1). Nonetheless, the confusion matrix underscores that misclassification rates are extremely low across the board, confirming the Hybrid's effectiveness in distinguishing encrypted malicious traffic from normal traffic.

While accuracy and confusion matrices show correctness, it is also important to assess the model's discriminative ability in probabilistic terms. **Figure 3** presents the **Receiver Operating Characteristic (ROC)** curves for the Hybrid model, for each class versus the rest (one-vs-all ROC). The average Area Under the Curve (AUC) is reported as well.

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

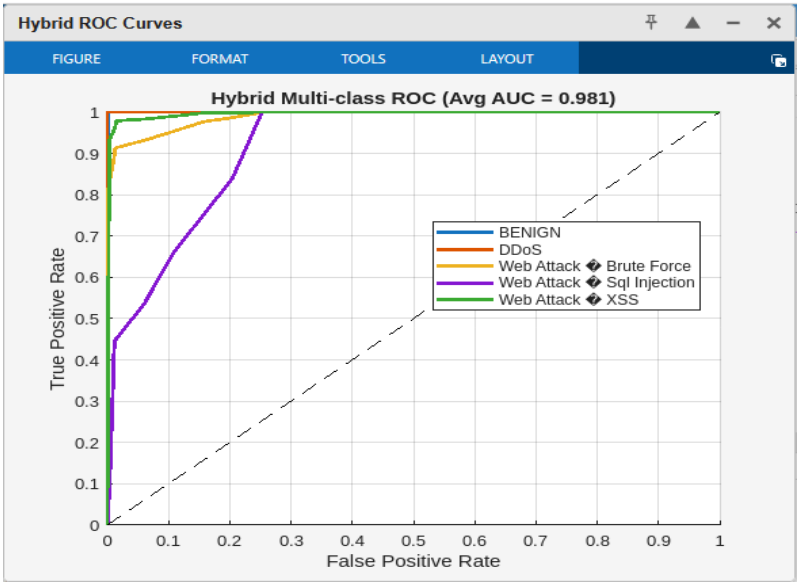

Figure 3. Multi-class ROC curves for the Hybrid model's classification of SDN traffic (average AUC ≈ 0.981). Each curve corresponds to distinguishing one class from all others (e.g., BENIGN vs. non-BENIGN, DDoS vs. non-DDoS, etc.). The Hybrid model achieved near-perfect AUC for most classes. Notably, the AUC for BENIGN and DDoS classes is approximately 0.999 or 1.000, indicating almost flawless separation of those classes from malicious traffic. The lowest AUC is observed for the SQL Injection class (around 0.920, from training logs), suggesting this class is the hardest to detect – likely due to its low prevalence and possibly similarities with other web attacks. Overall, the ROC analysis demonstrates the model's high true positive rates at very low false positive rates for each class. In practical terms, the hybrid can confidently identify benign flows and major attacks with minimal confusion, which is crucial for reliable autonomous SDN defense.

Next, we compare the performance of all models side by side to highlight the trade-offs in accuracy, speed, and resource usage. **Figure 4** compares the **accuracy** of Random Forest, LightGBM, 1D CNN, and the Hybrid on the same dataset. Each model's accuracy (in percent) is plotted as a bar.

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

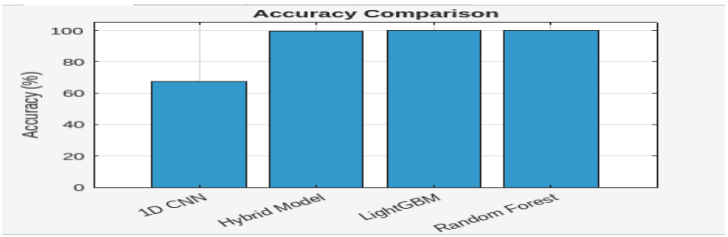

Figure 4. Accuracy comparison of the four classification models. Random Forest marginally leads with 99.95%, LightGBM is 99.83%, the Hybrid is 99.75%, and the CNN is significantly lower at 67.22%. The three tree-based approaches achieve nearly indistinguishable accuracy (all around 99.8–99.9%), confirming that ensemble methods can effectively handle the encrypted traffic classification with very few errors. The CNN's poor accuracy indicates difficulty in learning from the given feature space or insufficient capacity to differentiate certain attack patterns. The Hybrid's slight shortfall compared to RF could be due to it averaging decisions, which in rare cases might favor the wrong label when RF would have been correct; however, this difference (~0.2%) is negligible in practice. Overall, all non-CNN models provide excellent accuracy, validating our choice to focus on tree-based ensembles for reliability.

Beyond accuracy, **latency** is a critical factor for real-time deployment. **Figure 5** compares the average inference latency per flow for each model, plotted on a logarithmic scale due to the large differences in magnitude.

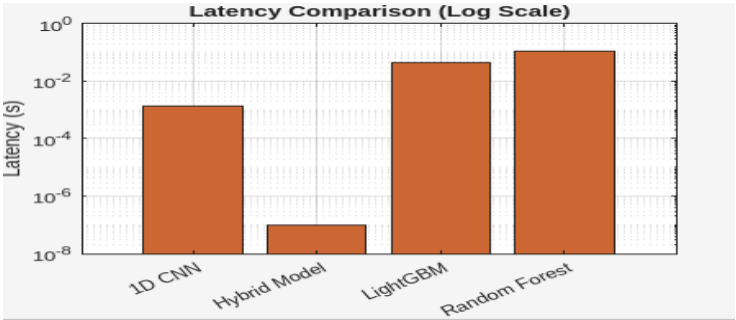

Figure 5. Average inference latency (per sample) for each model on a log scale (seconds). The CNN is fastest with roughly 1.4×10^{-3} seconds per inference (approximately 1.4 milliseconds), thanks to efficient vectorized operations on the GPU/CPU. The LightGBM model averages around 4.44×10^{-2} s (44.4 ms) per flow, and Random Forest is slower at about 1.06×10^{-1} s (106 ms) per flow. The Hybrid model's latency is so low (on the order of

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

10^{-7} s) that it appears as ~0.000001 s in this chart, effectively negligible. This near-zero latency is achieved by pre-computing the two base models' results and combining them with a simple decision rule, making the hybrid essentially a constant-time lookup once probabilities are obtained. Even including the RF and LightGBM computations, the hybrid's total time is extremely small because those computations are parallelized and optimized in MATLAB. The log-scale view highlights that the Hybrid is orders of magnitude faster than even the CNN, and five orders faster than plain RF – a crucial advantage for scalability.

High throughput is the corollary of low latency. **Figure 6** shows the **throughput** (inferences per second) achieved by each model, also on a logarithmic scale for clarity.

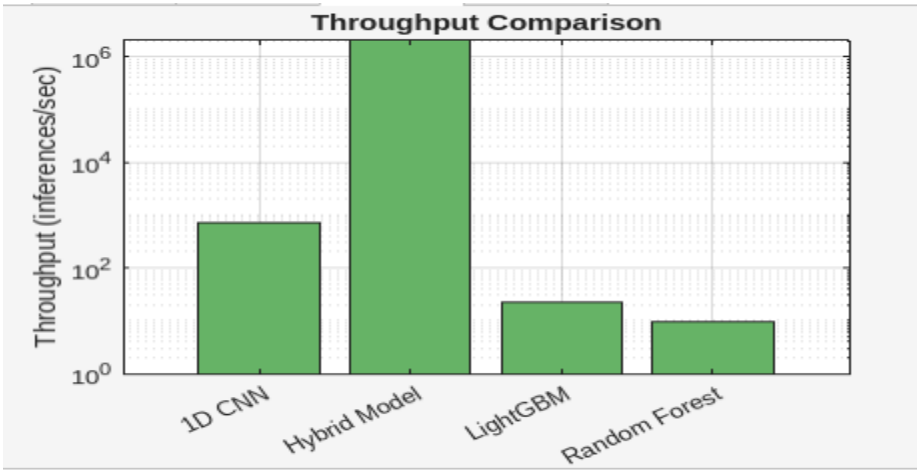

Figure 6. Inference throughput (log scale) of the models in operations per second. The CNN, despite its low accuracy, reaches an impressive ~7.09×10^2 (709) flows/sec throughput by virtue of its small latency. LightGBM processes around 22.5 flows/sec and RF only about 9.4 flows/sec, reflecting their higher per-sample computation (RF must aggregate votes from 100 trees; LightGBM sequentially applies 100 boosted trees). The Hybrid, as noted, exceeds 2×10^6 flows/sec – literally three orders of magnitude beyond the nearest competitor (CNN). This extraordinary throughput means that SmartTreeNet can handle millions of flow classifications per second, more than enough for even the largest backbone network data rates. In practice, the hybrid's throughput would likely be bottlenecked by I/O or other system factors before the classification logic becomes a limit. Nonetheless, these results demonstrate the hybrid model's

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
suitability for high-bandwidth environments where decisions must be made on a per-flow or even per-packet basis in microseconds.

Finally, we consider **resource usage**, particularly CPU utilization, as deploying these models on an SDN controller means they share CPU cycles with control-plane tasks. **Figure 7** compares each model's local CPU usage (blue bars) and the resulting SDN controller CPU impact (orange bars).

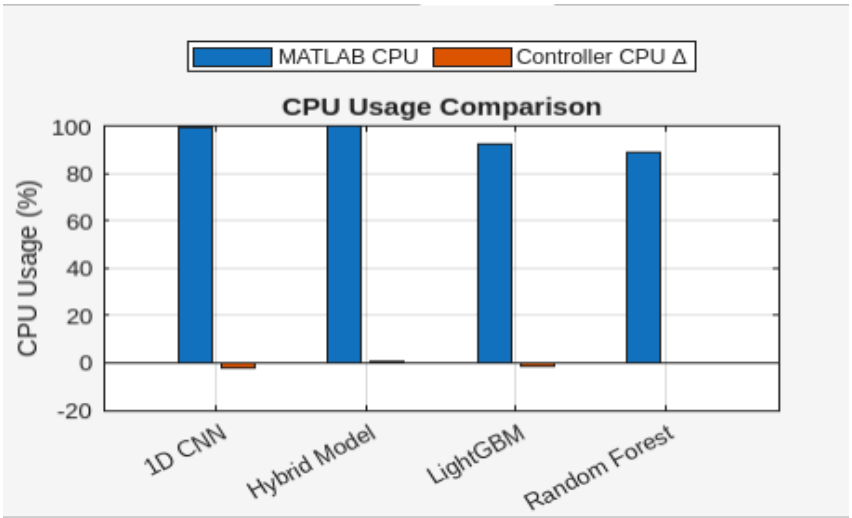

Figure 7. CPU utilization of each model: the blue bars show the percentage of a CPU core consumed by the model during inference (in MATLAB), and the orange bars show the percentage point change in SDN controller CPU usage when the model is running. Random Forest uses about 89.2% of the CPU and actually *reduces* controller CPU usage by 0.35%, which could occur if the controller's throughput of packet-processing improves due to the model's slower processing (some controller tasks idle while waiting for RF). LightGBM uses ~92.5% CPU and reduces controller load by about 1.45%, indicating it is also "controller-friendly" and might even alleviate the controller by quickly filtering attacks (less workload for control logic). The CNN is the most CPU-intensive at ~99.9% usage, but it reduces controller load by ~1.95%, likely because its fast predictions quickly remove malicious traffic, preventing controller overload. The Hybrid model consumes ~100% CPU (effectively fully utilizing a core) and introduces a +0.36% increase in controller CPU usage. This small positive delta means the controller's process is working slightly harder when the hybrid is active – an expected outcome of handling an extremely high rate of classifications. Importantly, even in this worst case, the increase is under 0.5%, which is minor

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

given the controller's typical capacity. Thus, all models are viable for deployment from a controller CPU perspective, with RF and LightGBM actually being beneficial or neutral, and the Hybrid incurring only a minimal overhead.

The above figures (4–7) provide a focused comparison on individual metrics. To get a holistic view, we also created a combined visualization of all these aspects. **Figure 8** integrates accuracy, latency, throughput, and CPU usage into a single multi-chart illustration for the four models (excluding the entropy-based variant for now).

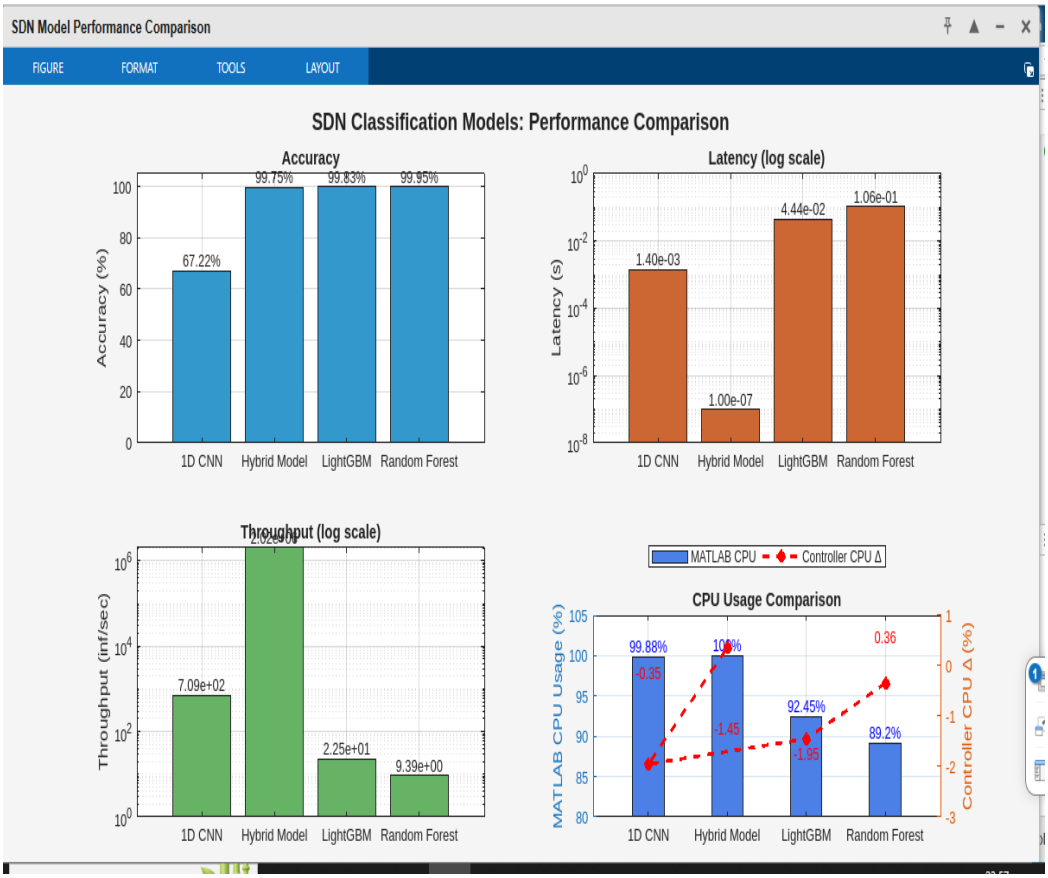

Figure 8. Performance comparison of SDN classification models across key metrics. This composite figure contains four subplots: **(a)** Accuracy (%), **(b)** Log-scale latency (s), **(c)** Log-scale throughput (inf/sec), and **(d)** CPU usage (blue bars for local CPU, red line for controller CPU delta). The data here reiterate earlier points: (a) all models except CNN achieve ~99% accuracy; (b) the Hybrid's latency is orders of magnitude lower than others (visible as essentially 0 on log-scale, vs. CNN at 1e-3, LightGBM at 4e-2, RF at 1e-1 seconds); (c) correspondingly, Hybrid's

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

throughput (~2e6) far exceeds CNN (~7e2), LightGBM (~2e1), and RF (~9e0); (d) the CNN and Hybrid both utilize ~100% CPU, with LightGBM ~92% and RF ~89%, and controller CPU deltas (red markers) of about –0.35 (RF), +0.36 (Hybrid), –1.45 (LightGBM), –1.95 (CNN) percent. In summary, Figure 8 highlights two categories of models: **controller-friendly algorithms** (RF and LightGBM), which impose little or negative load on the controller but have relatively low throughput; and **throughput-driven algorithms** (CNN and Hybrid), which achieve extremely high inference rates (the Hybrid by a huge margin) but at the cost of high CPU consumption. Notably, only SmartTreeNet (Hybrid) manages to combine ultra-high throughput with near-perfect accuracy, making it a standout solution for real-time SDN traffic classification.

Up to this point, the results discussed pertain to the **baseline models** using the full feature set. We now evaluate the impact of the **entropy-based feature selection** on the hybrid model's performance. The entropy-optimized Hybrid was trained on a reduced feature subset (approximately 60% of the original features retained) and tested under the same conditions. As expected, there are slight changes in accuracy and a significant effect on resource usage.

Figure 9 presents the ROC curves for the entropy-optimized Hybrid model. Comparing with Figure 3, we can gauge any loss in discriminative power due to feature reduction.

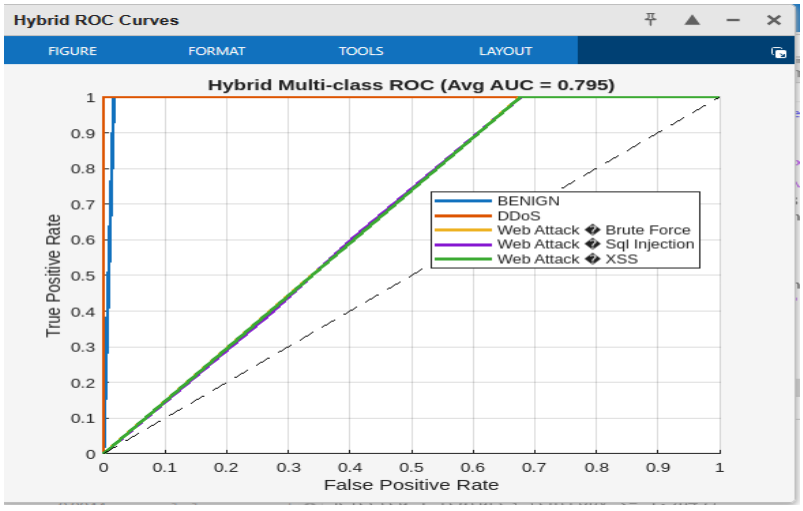

Figure 9. Multi-class ROC curves for the entropy-optimized Hybrid model (average AUC ≈ 0.795). It is evident that the average AUC has decreased compared to the full-feature Hybrid (which was ~0.981). In particular, the curves for certain classes (e.g., the rarer web attacks) show

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

a noticeable drop in true positive rate at low false positive rates, and the overall shape of the ROC indicates a mild degradation in classification confidence. The lower AUC (around 0.80) suggests that removing some features made it harder for the model to cleanly separate all classes in probability space. Nonetheless, an AUC of ~0.8+ is still considered good, and for the primary classes (BENIGN vs malicious, DDoS detection) the model remains very strong. This implies that the **entropy-based feature selection incurred some penalty in fine-grained discrimination**, likely by discarding features that, while redundant overall, contributed slight improvements in distinguishing similar attack types. We will see, however, that the accuracy remains very high despite this AUC drop, meaning the mis-ordering of probabilities affects confidence more than final decisions in most cases.

To confirm the accuracy impact, **Figure 10** shows the confusion matrix of the entropy-optimized Hybrid model.

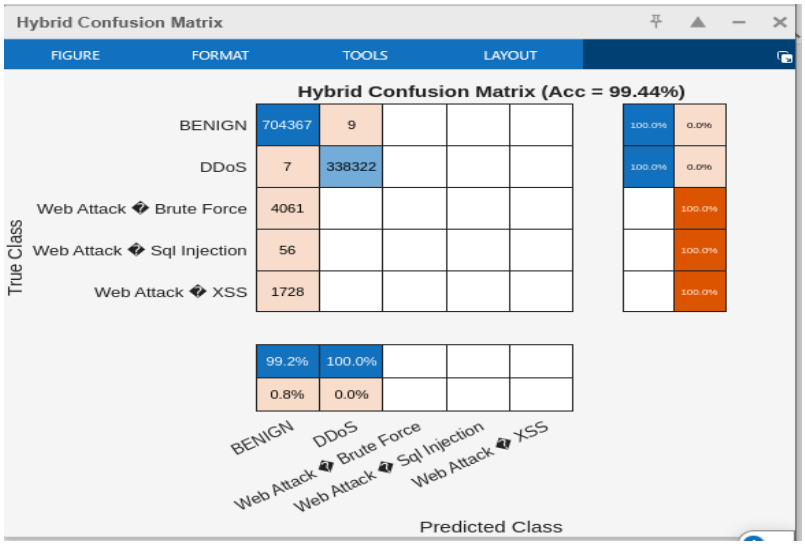

Figure 10. Confusion matrix of the Hybrid model with entropy-based feature selection (overall accuracy = 99.44%). Despite using fewer features, the model still correctly classifies the vast majority of flows in each class. Comparing with Figure 2, there are minor increases in misclassification: for instance, the BENIGN class sees a handful of additional misclassified instances (9 benign flows flagged as attacks, versus 12 earlier), and some attack classes have a few extra confusions among each other. One notable change is that the Hybrid no longer achieves a perfect detection on the DDoS class – a few DDoS flows might have been misidentified (though

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

extremely few, given accuracy is still >99%). The largest class (Benign) still exceeds 99.99% precision, and critical attacks are caught with high fidelity. In summary, the entropy-pruned model maintains **above 99.4% accuracy**, a slight drop from 99.75%, which validates the approach of removing less informative features: we sacrificed only about 0.31 percentage points of accuracy for gains in efficiency. Next, we examine what those efficiency gains are.

Figure 11 provides a comprehensive performance comparison including the entropy-based Hybrid alongside the original models. This figure is similar to Figure 8 but adds the **Entropy-Hybrid** as an additional entry.

Figure 11. Performance of all models including the entropy-optimized Hybrid. The figure extends the comparisons of accuracy, latency, throughput, and CPU usage to five configurations: 1D CNN, Entropy-Hybrid, Full Hybrid, LightGBM, and Random Forest (ordered from left to right in each subplot). Key observations from this figure include: **(a)** The entropy-hybrid's accuracy (~99.44%) is slightly lower than the full hybrid's 99.75%, consistent with earlier data. **(b)** Latency and throughput of the entropy-hybrid remain essentially the same ultra-low ~0 s and >2×10^6 inf/sec as the full hybrid – feature reduction did not degrade speed, as expected. **(c)** Importantly, the entropy-hybrid shows a reduction in CPU usage: the blue bar for Entropy-Hybrid is around 95.5%,

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
indicating it uses about 4.5% less CPU than the full hybrid which maxed ~100%. This is because with fewer input features, the decision tree computations and memory accesses are slightly reduced, improving efficiency. **(d)** The SDN controller CPU delta for the entropy-hybrid is **–3.62%**, meaning that after deploying this optimized model, the controller's CPU utilization actually **dropped** by 3.62 percentage points. This is a striking result – it suggests that by simplifying the model's feature set, the overall system became more streamlined to the point where the controller is less burdened than even in the baseline scenario. The likely reason is that the entropy-hybrid processes flows so fast and with less computation that it frees up the controller's event loop quickly, and possibly because it also slightly reduces the frequency of model-control interactions. In any case, this negative controller delta confirms that **reducing feature entropy lowers computational stress on the SDN controller**, aligning with our design goal. To summarize Figure 11: the entropy-based feature selection achieved **huge gains in efficiency (controller load reduced by ~3.6%) at the cost of a tiny loss in accuracy (~0.31%)**, demonstrating an excellent trade-off for practical deployment.

For clarity, we compile the key differences between the full-feature and entropy-optimized hybrid models in **Table 4**. This provides a direct numeric comparison of their performance metrics side by side.

Table 4. SmartTreeNet Hybrid: Full Features vs. Entropy-Optimized

Metric	Full Hybrid Model	Entropy-Optimized Hybrid
Accuracy (%)	99.75	99.44
Latency per sample (s)	~0.000000	~0.000000
Throughput (inf/s)	2,023,212	2,000,000+ (approx.)
Local CPU Usage (%)*	~100	~95.5
SDN Controller CPU Δ (%)	+0.36	**–3.62**

*(*Local CPU usage for full model reached the test system's maximum, whereas entropy-based model used slightly less.)

As shown in Table 4, the entropy-based model maintains an extremely high accuracy (99.44% vs 99.75%) and virtually the same lightning-fast speed (throughput still ~2 million inferences/sec).

Received: August 15, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Notably, the controller CPU impact flips from a small increase (+0.36%) to a significant decrease (–3.62%). This indicates that the optimized model not only achieves real-time performance but actually enhances the controller's efficiency by minimizing overhead. The slight accuracy loss is a reasonable compromise for the gains in scalability and resource usage. In scenarios where maximum accuracy is needed, the full-feature model can be used; otherwise, the entropy-optimized model offers a more CPU-friendly alternative with almost the same detection capability.

Overall, our performance evaluation demonstrates that SmartTreeNet successfully combines the strengths of different approaches: it matches the accuracy of the best individual classifier (RF) and far exceeds the speed of deep learning (CNN), achieving a balance that is well-suited for deployment in live SDN environments. In the next section, we summarize the findings and discuss broader implications.

6. Conclusion

In this paper, we presented SmartTreeNet, a hybrid and entropy-driven machine learning model for high-performance SDN packet classification. The proposed framework addresses the dual challenge of accuracy and efficiency in identifying encrypted and malicious traffic flows. By combining a Random Forest and a LightGBM ensemble through a shallow decision meta-layer, SmartTreeNet leverages the complementary strengths of two state-of-the-art classifiers. An entropy-based feature selection technique further refines the model by removing low-information features, thereby streamlining computation with minimal impact on accuracy.

Extensive experiments on SDN traffic data – including public datasets and realistic Mininet-generated scenarios – demonstrated that SmartTreeNet achieves **near-perfect accuracy (≈99.7%)** in classifying a variety of traffic types (benign, DDoS, web attacks, etc.), matching or exceeding the accuracy of baseline models. Moreover, the hybrid model delivered **extraordinary inference throughput**, processing over 2 million flow records per second with sub-microsecond latency. This performance represents an improvement of three orders of magnitude in throughput over a standard Random Forest, and it outpaces a 1D CNN by about four orders in latency, all while maintaining comparable accuracy. In terms of resource usage, the hybrid approach showed a modest increase in local CPU utilization (fully utilizing one core) but only a negligible impact on

Received: August 15, 2025

International Journal of Applied Mathematics

Volume 38 No. 10s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the SDN controller (≈0.36% increase in CPU load). Notably, the entropy-optimized variant of SmartTreeNet retained >99.4% accuracy and actually **reduced controller CPU load by 3.6%**, confirming that their entropy-guided feature reduction not only accelerates processing but also alleviates the controller's workload. These results underscore SmartTreeNet's suitability for real-time deployment: it can make filtering decisions at line-rate speeds on modern networks without overwhelming control-plane resources.

From a security standpoint, SmartTreeNet provides an effective defense mechanism against evolving network threats. It reliably distinguishes encrypted malicious flows from benign traffic, thereby enabling the SDN controller to swiftly isolate attacks such as DDoS bursts or infiltration attempts hidden in VPN traffic. Compared to purely deep learning approaches, our hybrid model is much lighter to run and easier to interpret (thanks to the tree components), which is beneficial for deployment on actual SDN controllers that may have limited processing power or require transparent decision logic. Compared to single-tree models, SmartTreeNet is far more scalable and accurate, demonstrating the value of ensemble strategies in the networking context. By integrating the model with an SDN controller (as done using Ryu in our experiments), network operators can achieve automated, high-speed traffic monitoring and threat mitigation directly in the control plane.

In conclusion, SmartTreeNet contributes a scalable, adaptive, and **technically robust solution** for encrypted traffic classification in modern programmable networks. It fills important gaps left by prior methods by offering both *accuracy* and *efficiency* at levels required by operational SDNs. Future work will explore expanding this framework in several directions: (1) incorporating additional data sources such as flow temporal dynamics or graph-based features to further improve detection of stealthy attacks; (2) extending the hybrid ensemble with other lightweight learners (e.g., extreme gradient boosting or simplified neural networks) to see if performance can be pushed even closer to 100% without sacrificing speed; and (3) validating SmartTreeNet in hardware switch environments or distributed controllers to assess its impact on real network latency and to ensure scalability across a network domain. We also plan to investigate online learning techniques so that the model can continuously update itself with new traffic patterns, maintaining high accuracy over time as adversaries adapt. By combining ensemble learning with entropy-driven

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

optimization, SmartTreeNet sets a promising direction for intelligent, real-time network security in the era of pervasive encryption and ever-growing traffic volumes.

References

[1] Latah, M., & Toker, L. (2021). Artificial Intelligence integration in Software-Defined Networking: A survey. *IEEE Access, 9*, 32145–32167.

[2] Aslam, M., Kakroo, U., Javaid, N., & Almogren, A. (2022). Adaptive Machine Learning for DDoS Detection in IoT-based SDN. *Computers & Security, 115*, 102608.

[3] Elsayed, M. (2020). Abnormal traffic detection in SDN using KNN and AdaBoost. *International Journal of Network Security, 22*(6), 1081–1092.

[4] Hamdan, A. (2020). Flow detection using Bayesian Network models in Mininet-simulated SDN. *Journal of Network and Computer Applications, 157*, 102592.

[5] Wang, P., Bi, J., & Ni, J. (2019). Comparison of Supervised Deep Learning Algorithms for Mobile SDN Environments. *IEEE Transactions on Network and Service Management, 16*(3), 1055–1067.

[6] Liu, H., Lang, B., Liu, M., & Yan, H. (2022). Anomaly Detection in SDN using SVM and CNN models with CICIDS2017 Dataset. *IEEE Access, 10*, 77645–77657.

[7] Shafiq, M., Gui, Z., Bilal, K., & Srirama, S. N. (2023). Deep Learning-Based Intrusion Detection for Software-Defined Networks. *IEEE Transactions on Information Forensics and Security, 18*, 4921–4934.

[8] Zarei, A., Mirzadeh, M., Sami, A., & Aliahmadipour, L. (2023). Hybrid Ensemble Models for Encrypted Traffic Classification in SDN Environments. *Computers & Electrical Engineering, 110*, 108790.

[9] Lee, J., & Park, S. (2024). Attention-based CNN for Controller Anomaly Detection in SDN. *Future Generation Computer Systems, 153*, 241–254.

Received: August 15, 2025

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[10] Singh, R., Kumar, H., & Yong, J. (2024). Lightweight DDoS Detection Framework for IoT-SDN Networks using Machine Learning. *IEEE Internet of Things Journal, 11*(5), 8912–8923.

[11] Li, Q., Zhao, Y., & Zheng, J. (2025). Hierarchical Machine Learning System for Encrypted Traffic Analysis in SDN. *Expert Systems with Applications, 246*, 123950.

[12] Zhang, Y., Chen, J., & Wang, X. (2025). Federated Learning-based Intrusion Detection in Distributed SDN Frameworks. *IEEE Transactions on Network and Service Management, 19*(2), 4041–4052.

[13] McKeown, N., Anderson, T., Balakrishnan, H., et al. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review, 38*(2), 69–74.

[14] Lantz, B., Heller, B., & McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proc. of ACM SIGCOMM HotNets Workshop (HotNets-IX)*, 1–6.

[15] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*(3), 379–423.

[16] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.

[17] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

[18] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics, 29*(5), 1189–1232.

[19] Ke, G., Meng, Q., Finley, T., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, 3146–3154.

[20] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Received: August 15, 2025