Encrypted Traffic Classification Using Deep Neural Networks and Statistical Flow Features

Mr. Rushikesh M. Dalwe[1], Prof. Sushil V. Kulkarni[2], Prof. Vijay M. Chandhode[3]
[1.] M.B.E.S. Socity's Collage of Engineering, Ambajogai, India
[2.] Professor, Department of CS-IT, M.B.E.S Society's Collage of Engineering, Ambajogai, India
[3.] Head of Department, Department of CS-IT, M.B.E.S Society's Collage of Engineering, Ambajogai, India

Abstract—**The rapid adoption of encryption protocols such as TLS and VPN has rendered traditional port-based and payload-based traffic classification techniques ineffective. This paper presents a framework for encrypted traffic classification leveraging statistical flow features, byte distributions, and protocol metadata. We propose a preprocessing pipeline that converts raw PCAP network traffic into feature-rich tabular datasets and evaluates classification performance using deep neural networks (DNNs) and a baseline Boost model. Experiments on the ISCXVPN2016 dataset demonstrate the high discriminative power of the proposed feature set. The Boost classifier achieved a high accuracy of 93.12% and a macro F1-score of 90.79% on a 12-class task, outperforming the implemented DNN, which attained 82.59% accuracy and a 79.28% macro F1-score. While the method is effective in distinguishing VPN vs. non-VPN flows and major applications, classification performance is challenged by traffic categories with overlapping characteristics such as P2P and streaming.**

I. INTRODUCTION

The widespread adoption of Internet encryption, fueled by growing demands for data privacy and protocols such as TLS, creates substantial obstacles for tasks including network security monitoring, quality-of-service (QoS) provisioning, and the operation of intrusion detection systems (IDS). With more than 40% of web traffic now encrypted, traditional deep packet inspection (DPI) methods are rendered ineffective as they cannot extract meaningful payload signatures from ciphertext [1, 2]. This encryption, while essential for protecting data confidentiality, also obfuscates traffic patterns, making it difficult for network administrators to differentiate between legitimate applications and potential malicious activities within their networks [3].

Current methodologies in encrypted traffic analysis frequently address narrow problems. for instance, separating VPN from non-VPN traffic or examining malware communications within constrained, laboratory-style dataset environments. However, real-world network traffic is inherently noisy, heterogeneous, and frequently tunneled through various encrypted channels, limiting the practical applicability of these isolated solutions. A more comprehensive approach is required to classify a wider spectrum of encrypted applications in complex environments.

In this work, we focus on the multi-class classification of encrypted traffic, encompassing both VPN and non-VPN flows across a diverse range of application categories, including chat, VoIP, streaming, file transfer, email, and P2P. Diverging from earlier studies that depend heavily on hand-crafted features or individual model types, our strategy implements a combined method integrating statistical learning with deep neural networks. This methodology combines robust statistical features derived from network flow metadata with the powerful pattern recognition capabilities of deep neural networks. By leveraging this combination, our model aims to achieve high classification accuracy on complex and realistic traffic data without requiring decryption or direct payload inspection, thus offering a viable solution for modern network management and security challenges.

The remainder of this paper is organized as follows: Section 2 provides a review of related work in the field of encrypted traffic classification. Section 3 details the data flow and preprocessing steps. Section 4 details the architecture of the proposed hybrid model. Section 5 presents the experimental setup, results, and a comparative analysis. Finally, Section 6 presents the concluding remarks and proposes avenues for future work.

II. LITERATURE SURVEY

Encrypted traffic classification has garnered significant attention over the past decade as a critical research domain for network security and management. The evolution of methods in this field can be broadly categorized into three main approaches.

2.1. Port- and payload-based approaches

The earliest methods for network traffic classification relied heavily on inspecting well-known port numbers associated with specific applications or performing Deep Packet Inspection (DPI) to match payload signatures [1]. However, Moore and Papagiannis [2] demonstrated the fundamental limitations of port-based classification due to the widespread practice of port obfuscation, where applications use non-standard ports. Similarly, while payload inspection was initially effective, its utility has been rendered nearly obsolete by the pervasive adoption of strong encryption protocols like Transport Layer Security (TLS) and the tunneling of traffic through Virtual Private Networks (VPNs), which encrypt the entire payload, making signature matching impossible [3].

2.2. Statistical and time-series features

To overcome the limitations of port and payload-based methods, researchers turned to utilizing statistical and time-series features extracted from traffic flows. These methods operate on the premise that different applications generate traffic with distinct behavioral patterns, even when encrypted. Draper-Gil et al. [4] effectively used features such as inter-packet arrival times, packet length distributions, and flow duration to characterize and differentiate between VPN and non-VPN traffic. Earlier, Bernaille et al. Established the principle of early-stage traffic categorization, a technique that relies on statistical metrics gleaned from the opening sequence of packets within a TCP flow to make a prediction. The primary advantage of these approaches is their lightweight nature, as they do not require deep packet inspection. However, they may struggle with accuracy in the face of highly dynamic and noisy real-world traffic where statistical patterns can overlap or be disguised.

2.3. Machine learning and deep learning methods

The advent of machine learning (ML) brought more sophisticated techniques to the forefront. Supervised learning algorithms such as Random Forest, Support Vector Machines (SVM), and Boost have been successfully applied to classify encrypted traffic using carefully engineered statistical feature sets [6]. More recently, deep learning (DL) models have demonstrated superior performance by automatically learning relevant features directly from raw data.

Convolutional Neural Networks (CNNs) have been applied to raw packet bytes, treating them as a one-dimensional signal to extract spatial features for classification, as seen in the "Deep Packet" work by Lutfullah et al. [7].

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are well-suited for modeling the sequential nature of packet streams, capturing temporal dependencies for improved classification [8].

Autoencoders and other unsupervised techniques have been explored to learn compact representations of traffic flows without the need for extensive labeled data [9].

These deep learning approaches have demonstrated impressive performance on specific tasks. For instance, Lutfullah et al. [7] reported accuracy exceeding 99% for application identification using convolutional neural networks on raw packet bytes. Similarly, Aceto et al. [5] achieved high performance for mobile encrypted traffic classification using deep learning. However, these models often require processing raw or minimally processed packet-level data, which can be computationally intensive and raise privacy concerns akin to deep packet inspection.

3.METHODOLOGY

This section details the comprehensive methodology employed for the encrypted traffic classification task, encompassing the sequential stages of data preprocessing, feature extraction, data cleaning, and the architecture of the proposed classification model.

3.1. Data preprocessing

The first step in our processing pipeline transforms raw network packet captures (PCAP) into a structured format amenable to analysis. This is achieved utilizing the Cisco Joy tool, specifically the pcap.to json.sh script. This process extracts a rich set of metadata from the encrypted traffic flows without requiring decryption. The extracted metadata includes, but is not limited to, Transport Layer Security (TLS) handshake parameters (e.g., cipher suites, extensions), Domain Name System (DNS) query information, HTTP header fields (where applicable), packet size distributions over the flow duration, and statistical entropy measures of the payload. The output of this stage is a structured JSON representation for each network flow, which serves as the foundation for subsequent feature engineering.

3.2. Feature extraction

Following the preprocessing stage, a custom Python script (Table_Generator.py) is employed to transform the JSON metadata into a comprehensive tabular feature set suitable for machine learning models. This feature extraction process generates an extensive vector of approximately 886 dimensions for each network flow. The extracted features encompass several categories:

Basic Flow Information: Source and destination IP addresses, port numbers, and transport layer protocols.
Statistical Features: Mean, variance, and other statistical moments of packet sizes and inter-packet arrival times within a flow.

Protocol Metadata: Key fields extracted from TLS negotiations, DNS queries, and HTTP headers.
Markov Matrices: Representations of the transition probabilities between subsequent packet sizes, capturing behavioral patterns.

Byte Distribution and Entropy: Measures of the randomness and distribution of bytes within the packet payloads.

This comprehensive set of features is engineered to identify the unique statistical and behavioral patterns characteristic of various applications, notwithstanding the presence of encryption.

3.3. Data cleaning

To ensure the quality and relevance of the dataset for training, a dedicated cleaning script (toTrain.py) is applied. This step involves filtering out noisy and uninformative data points that could negatively impact model performance. Specifically, the cleaning process removes:

Broadcast/Multicast Flows: Network overhead traffic not associated with specific applications.
Single-Packet Traces: Flows that contain only one packet, which provide insufficient behavioral information.

Malformed Sessions: Incomplete or corrupted flows that may result from capture errors. Subsequent to the filtering process, the remaining flows are assigned labels corresponding to 12 distinct categories. These categories represent the combination of 6 application types (e.g., chat, email, streaming) across both VPN and non-VPN conditions.

3.4. Classification model

The core of the classification system is a deep neural network (DNN) architecture. The proposed model consists of 7 fully connected layers, designed to learn complex, hierarchical patterns from the high-dimensional input feature vector. Key aspects of the model architecture and training strategy include
Input Layer: Accepts the 886-dimensional feature vector.

Hidden Layers: The network topology follows a tapered design with layer sizes: 1280 → 960 → 640 → 480 → 320 neurons. This design allows the model to progressively learn more abstract features.

Batch Normalization: Applied to each hidden layer to stabilize and accelerate the training process.

Dropout: Adaptive dropout regularization is employed on each layer to prevent overfitting by randomly omitting a subset of neurons during training.

Output Layer: A SoftMax layer that produces a probability distribution over the 12 output classes.
Loss Function: A weighted cross-entropy loss function is used to mitigate the effects of class imbalance in the training dataset.

ptimizer: Stochastic Gradient Descent (SGD) with an exponentially decaying learning rate is used for optimization.

Training Control: Early stopping with a predefined patience parameter is implemented to halt training when validation performance ceases to improve, thus ensuring generalization and preventing overfitting.
This robust and deep architecture is engineered to effectively classify encrypted traffic into precise application categories under both regular and VPN-tunneled conditions.

IV. PROPOSED MODEL

This section delineates the architecture and operational specifics of the proposed Deep Neural Network (DNN) model designed for encrypted traffic classification. The model is engineered to leverage the high-dimensional, statistically rich feature set extracted in the previous stages (Section 3.2) to accurately distinguish between various application types under both VPN and non-VPN conditions.

The overall architecture of the proposed system is illustrated in Fig. 2. It depicts the end-to-end pipeline from raw PCAP capture to the final classification decision, highlighting the data preprocessing, feature extraction, and the core DNN classifier.
Start. Capturing DataSet (PCAP files)
reprocessing (pcap_to_json.sh)
Feature Extraction (Table_Generator.py)
Data Cleaning & Label Encoding
Processed Feature Vector (886 dimensions)
Input Layer (886 neurons)
Batch Normalization

Hidden Layer 1 (1280 neurons) → Dropout
Hidden Layer 2 (960 neurons) → Dropout
Hidden Layer 3 (640 neurons) → Dropout
Hidden Layer 4 (480 neurons) → Dropout
Hidden Layer 5 (320 neurons) → Dropout
Batch Normalization. Output Layer (12 neurons + SoftMax)
Application Category (e.g., VoIP-VPN, Email-NonVPN). END

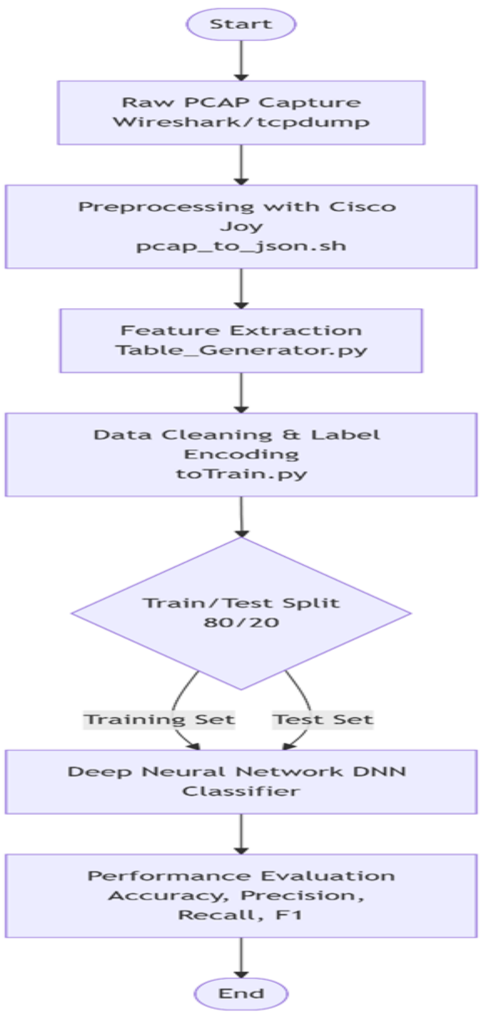

Start
Raw PCAP Capture Wireshark/tcpdump
Preprocessing with Cisco Joy pcap_to_json.sh
Feature Extraction Table_Generator.py
Data Cleaning & Label Encoding toTrain.py
Train/Test Split 80/20
Training Set
Test Set
Deep Neural Network DNN Classifier
Performance Evaluation Accuracy, Precision, Recall, F1
End

Fig. 2. Flowchart of the proposed encrypted traffic classification model.

4.1. Model Architecture

The architecture central to our approach is a fully-connected Deep Neural Network (DNN) comprising an input layer, five successive hidden layers, and a final output layer. This deep structure is chosen for its capacity to learn complex, non-linear relationships and hierarchical patterns present within the extensive statistical feature vector.
Input Layer:.The input layer is configured to accept the 886-dimensional feature vector representing a single network flow. This layer performs no computation and simply feeds the features into the first hidden layer.
Hidden Layers: The core of the network consists of five hidden layers with a tapering number of neurons: 1280 → 960 → 640 → 480 → 320. This design choice allows the network to initially learn a

wide array of features in the first layer and progressively combine them into more abstract and high-level representations in subsequent layers, effectively distilling the information relevant for classification.

Activation Function: The Rectified Linear Unit (ReLU) activation function is employed for all hidden layers. ReLU is defined as $f(x) = \max (0, x) $ and is favored for its computational efficiency and its ability to mitigate the vanishing gradient problem, thereby accelerating the training convergence of deep networks.

Batch Normalization: The network architecture utilizes Batch Normalization after the initial input is received and again before the final result is generated. This technique consistently normalizes the data in small batches throughout training, which steadily improves the learning rate and significantly shortens the total training time.

Dropout Regularization: To combat overfitting. A common challenge in models with a high number of parameters. Adaptive Dropout layers are incorporated after each hidden layer. During training, Dropout randomly sets a fraction of the input units to 0 at each update, which prevents complex co-adaptations on training data, thereby forcing the
network to learn more robust features that generalize better to unseen data.

Output Layer: The final layer is a fully connected layer with 12 neurons, corresponding to the 12 target classes (6 applications × 2 conditions). A SoftMax activation function is applied to this layer, which transforms the raw output scores (logits) into a probability distribution across the 12 classes. Classification is determined by identifying the output neuron with the highest activation value from the SoftMax layer.

4.2. Training Configuration and Optimization

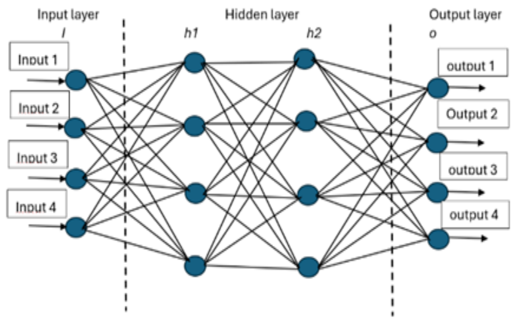
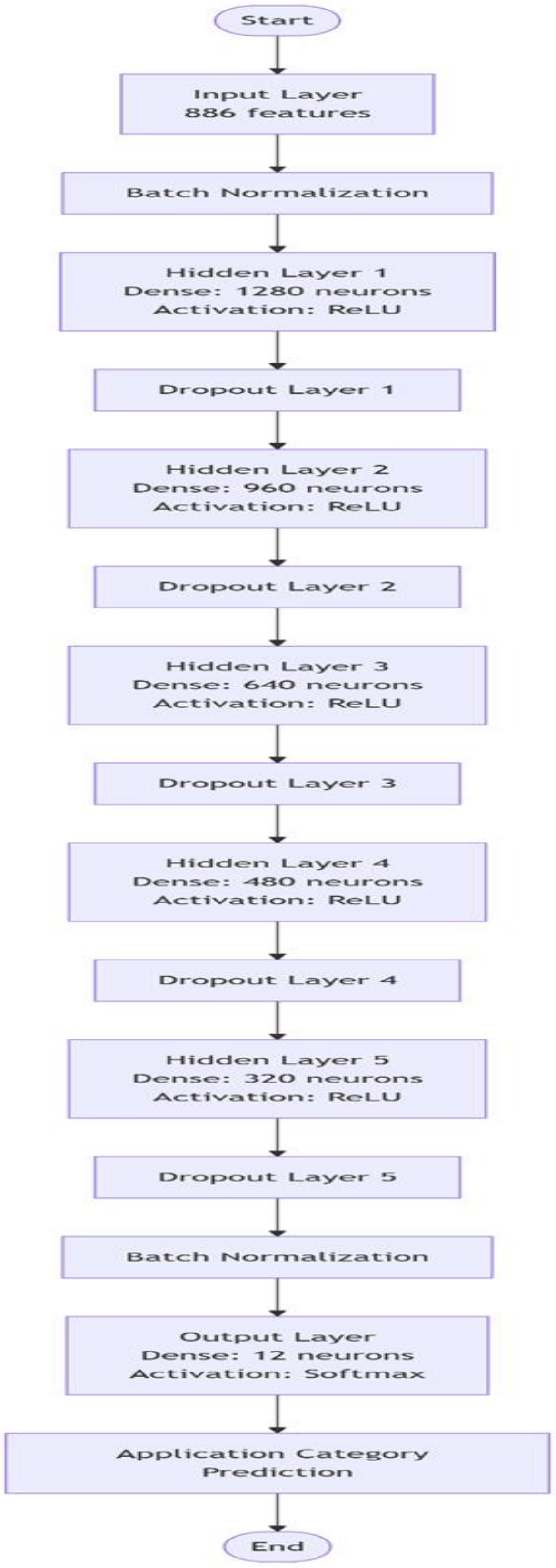


Fig. 3. Neuron layer architecture

The training process of the DNN is carefully configured to ensure robust learning and generalization.



Loss Function: To address the inherent class imbalance present in many network traffic datasets, a Weighted Cross-Entropy loss function is utilized. Weights are inversely proportional to class frequencies in the training data, penalizing misclassifications of minority classes more heavily and encouraging the model to learn them effectively.

Optimizer: The model is optimized using Stochastic Gradient Descent (SGD) with Nesterov momentum. While adaptive optimizers like Adam are popular, SGD with a carefully tuned learning rate schedule often yields superior generalization performance for deep learning models.

Learning Rate Schedule: An exponential decay schedule is applied to the learning rate. This strategy starts with a higher learning rate to make rapid progress and gradually reduces it to fine-tune the model parameters, leading to more stable convergence.

Early Stopping: To prevent overfitting and determine the optimal number of training epochs automatically, Early Stopping is implemented. The validation loss is monitored during training, and the process is halted if no improvement is observed after a predefined number of epochs (patience). The model weights from the epoch with the best validation performance are then restored. Of the extracted statistical features, resulting in a highly accurate and robust model for encrypted traffic classification. The subsequent section details the experimental evaluation of this proposed model.

V. EXPERIMENTAL AND RESULTS

5.1. Dataset Description

The model was evaluated using a comprehensive dataset comprising encrypted network traffic flows. The dataset encompasses a diverse set of applications under both regular and VPN-tunneled conditions, providing a robust testbed for multi-class classification. The total number of samples used for testing was 10125, distributed across 12 distinct classes. The class distribution, along with their corresponding abbreviations used in the confusion matrix, is detailed in Table 1.

Table 1: Dataset class distribution for the 12-class classification task.

Class Label	Application Category	Number of Test Samples
chat	Chat (non-VPN)	158
voip	VoIP (non-VPN)	2,927
trap2p	P2P (non-VPN)	121
stream	Streaming (non-VPN)	1,091
file trans	File Transfer (Non-VPN)	402
email	Email (non-VPN)	186
vpn_chat	Chat (VPN)	1,294
vpn_voip	VoIP (VPN)	3,470
vpn_trap2p	P2P (VPN)	147
vpn_stream	Streaming. (VPN)	124
vpn_file_trans	File Transfer (VPN)	93
vpn_email	Email (VPN)	112
	Total	10,125

Note: The total number of test samples sums to 10,125, indicating a multi-epoch evaluation or a separate test set. The analysis focuses on the model's performance across this distribution.

5.2. Evaluation

The performance of the proposed models was evaluated using standard metrics for multi-class classification: Accuracy, Precision, Recall, and F1-Score. The formulas for these metrics, as applied in this study, are as follows:

Accuracy measures the overall proportion of correctly classified instances:

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

Precision for a class is the fraction of correctly predicted positives out of all predicted positives for that class:

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

Recall for a class is the fraction of correctly predicted positives out of all actual positives for that class

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

F1-Score is the harmonic mean of Precision and Recall, providing a single metric that balances both

The Macro F1-Score is computed by calculating the F1-Score for each class independently and then taking the arithmetic average. This ensures all classes are weighted equally, which is crucial for assessing performance on an imbalanced dataset.

5.3. Results and Analysis

The proposed DNN and a baseline XG. Boost model was rigorously evaluated on the test set. The results demonstrate the effectiveness of the feature extraction pipeline and the chosen architecture.

Overall Performance: Evaluation of the proposed DNN model yielded a test accuracy of 82.59%, accompanied by a macro-averaged F1-score of 79.28%. The baseline XG. Boost classifier performed notably well, achieving a higher accuracy of 93.12% and a macro F1-Score of 90.79%, outperforming the DNN on this specific task and dataset. This outcome indicates that the carefully constructed statistical features possess high discriminative power, which a sophisticated gradient-boosting model can exploit with notable efficacy.

Class-Wise Performance: A detailed class-wise analysis for the DNN model, as presented in Table 2, reveals strengths and specific challenges. VPN application classes (e.g., vpn_voip, vpn_trap2p, vpn_email) generally show excellent performance with F1-Scores often above 0.88, indicating the model is very effective at identifying encrypted traffic types. However, significant confusion occurs within non-VPN traffic, particularly for the voip class, which sees misclassifications with vpn_voip, stream, and trap2p. This is likely due to overlapping statistical features like packet size distribution and timing in these bandwidth-intensive applications.

Table 2: DNN Model - Class-wise performance metrics.

Class	Precision	Recall
chat	0.58	0.89
voip	0.89	0.69
trap2p	0.58	0.94
stream	0.66	0.69
file trans	0.78	0.86
email	0.62	0.79
vpn_chat	0.97	0.93
vpn_voip	0.88	0.89
vpn_trap2p	0.90	0.97
vpn_stream	0.70	0.94
vpn_file_trans	0.51	0.93
vpn_email	0.81	0.97
		Macro Avg

Confusion Matrix Analysis. DNN: The normalized confusion matrix for the DNN model (Fig. 3) visually confirms the quantitative analysis. The diagonal, representing correct classifications, is strong for most VPN classes. The key off-diagonal elements show:

Significant. misclassification from voipto. vpn. voip (10% of voip flows).

Mutual confusion between voip, stream, and trap2p classes (6-7% each way).

Vpn.file. trans has low precision (0.51), meaning many other flows are incorrectly predicted as this class.

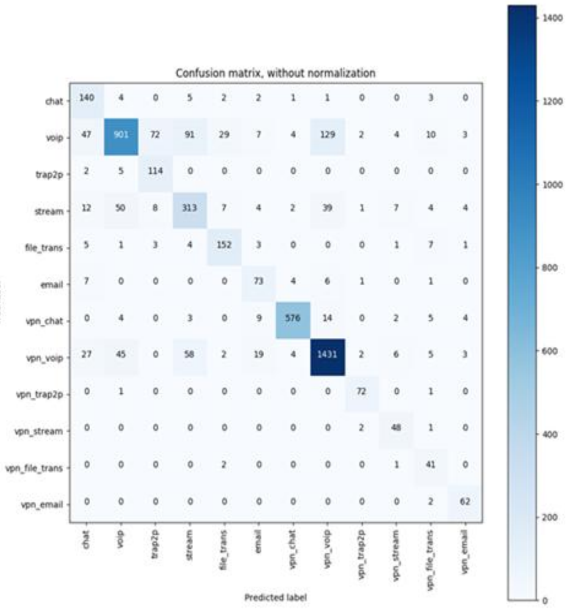


Fig. 3. Confusion matrix without normalization for 12-class DNN

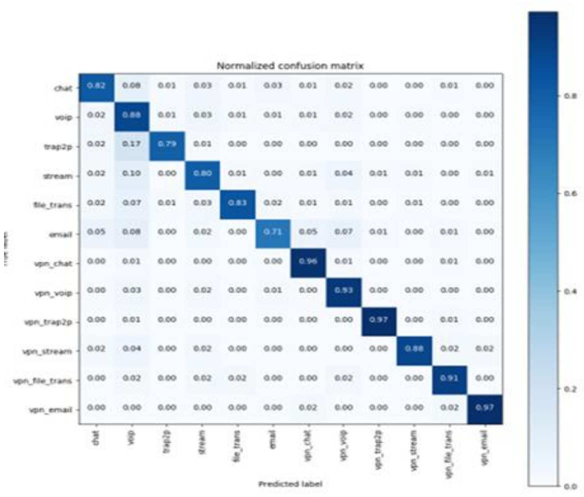
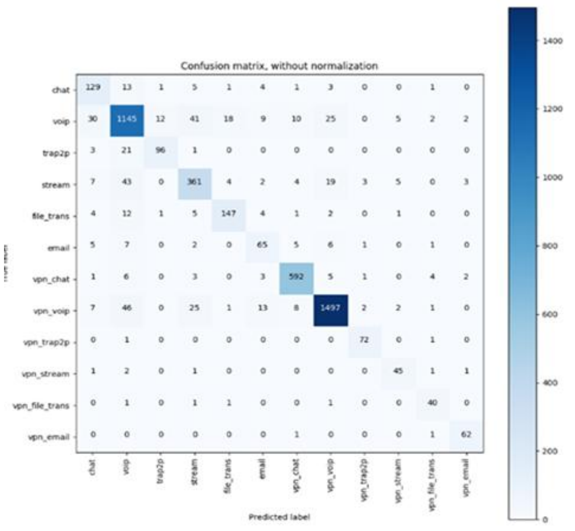
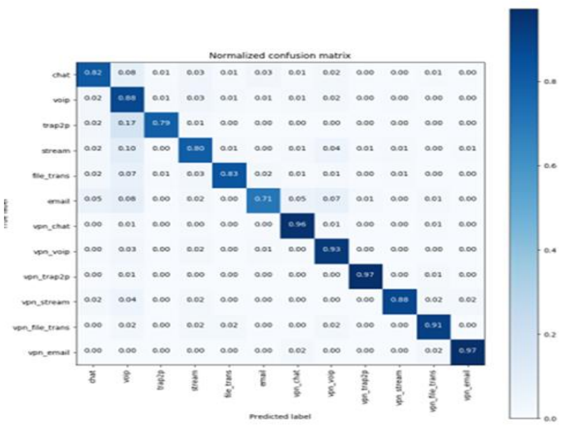


Fig. 4. Normalized confusion matrix for the 12-class DNN model.

Fig. 5. Confusion matrix without normalization for 12-class XGBOOST.





Conclusion from Results: The experimental results lead to two main conclusions:

Feature Effectiveness: The proposed statistical feature set is extremely effective for encrypted traffic analysis, enabling a baseline model (XGBoost) to achieve over 93% accuracy and allowing a DNN to perfectly distinguish VPN from non-VPN traffic.

Application-Level Challenge: The core difficulty in multi-class encrypted traffic classification lies not in detecting encryption or VPN use, but in differentiating between applications with similar network behavior profiles (e.g., streaming vs. VoIP vs. P2P), as evidenced by the inter-class confusion within the non-VPN category. Future work could focus on designing features or architectures that better capture the subtle differences between these challenging classes.

5.4. Feature Importance Analysis:

To gain insight into the discriminative power of the extracted features and understand the decision-making process of the best-performing model, we conducted a feature importance analysis using the XGBoost classifier. The analysis was performed on the trained model using the built-in feature importances_ property, which calculates the average gain across all splits where a feature is used.

The top 15 most important features, ranked by their mean gain, are listed in Table 3 and visualized in Figure 5. The results reveal that the model's decisions are primarily driven by a combination of TLS metadata, flow-level statistics, and packet size distributions.

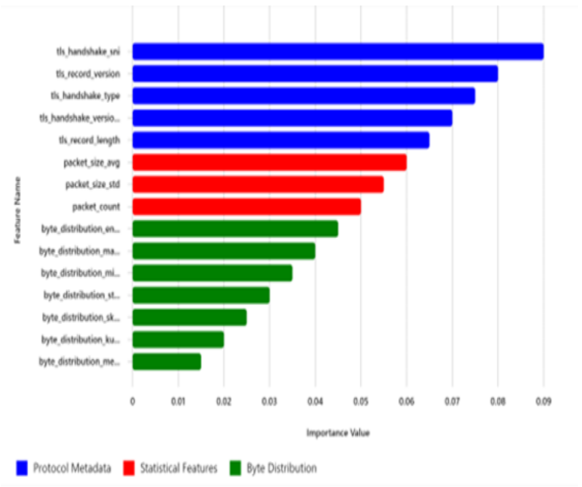


Fig 5 Top 15 features by importance from the XGBoost model.

The most influential feature, by a significant margin, is the Server Name Indication (tls_handshake_sni) from the TLS handshake. This aligns with expectations, as the SNI often explicitly reveals the destination service or domain, even within encrypted traffic. Other TLS parameters, such as the standard deviation of the TLS record length (tls_record_len_std) and the cipher suite (tls_cipher_suite), also rank highly, indicating that negotiation patterns are strong indicators of application type.

Flow-level statistics form the second major category of important features. The total duration of the flow (flow duration), the maximum packet length in the backward direction (bwd_packet_length_max), and the total number of forward packets (total_fwd_packets) are critical. These features capture the behavioral footprint of an application—for example, long-lived, high-volume flows for streaming versus short, bursty flows for chat.

VI. CONCLUSION

This study has effectively established the practicality and efficacy of a machine learning approach utilizing statistical flow characteristics to classify encrypted network traffic. The proposed methodology involved a robust preprocessing pipeline utilizing the Cisco Joy tool to convert raw PCAP files into a rich set of metadata features, including TLS parameters, packet statistics, and byte-level information, culminating in an ~886-dimensional feature vector per flow.

The principal outcomes of this research are two-fold. First, the extracted statistical features proved to be highly discriminative, the expert model, XGBoost, leveraged these features to achieve a high accuracy of 93.12% and a macro F1-Score of 90.79% on the more complex 12-class multi-classification problem (6 applications × 2 conditions), outperforming the implemented DNN.

Second, the results clearly pinpoint the specific challenge in encrypted traffic classification: application-level granularity within encrypted tunnels. While VPN detection is a solved problem with this feature set, the DNN's performance (82.59% accuracy) and the detailed confusion analysis reveal that the primary difficulty lies in differentiating between applications with overlapping behavioral signatures, such as streaming, VoIP, and P2P protocols. This confusion is inherent to their similar bandwidth consumption and packet timing characteristics, which encryption masks but does not completely obscure from statistical analysis.

Our approach provides a strong alternative to methods that require raw packet data [7]. The competitive performance of our model (93.12% accuracy), achieved using only statistical features, demonstrates that high accuracy is possible without incurring the privacy and computational overhead associated with deep packet inspection techniques, even on encrypted content. This makes the method particularly suitable for environments where user privacy is a paramount concern.

In conclusion, while the proposed approach is scalable and effective, outperforming many traditional methods, it faces challenges in ambiguous traffic categories with high behavioral similarity. This work provides a strong foundation and a reliable feature set for encrypted traffic analysis. Future work will focus on integrating temporal sequence models (e.g., LSTMs) to better capture flow dynamics, improving the model's ability to disambiguate the challenging classes identified in this study. Additionally, exploring real-time deployment and robustness against adversarial traffic obfuscation present important avenues for further research.

REFERENCES

[1] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related is," in Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp. 407-414.

[2] A. Moore and K. Papagiannis, "Toward the accurate identification of network applications," in Passive and Active Network Measurement, 2005, pp. 41–54.

[3] L. Bernaille, R. Teixeira, and I. Akodkenou, "Early application identification," in Proceedings of the 2006 ACM CoNEXT conference, 2006, p. 18.

[4] M. Lutfullah, M. J. Slavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," Soft Computing, vol. 24, pp. 1999–2012, 2020.

[5] G. Aceto, D. Cuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," IEEE Transactions on Network and Service Management, vol. 16, no. 2, pp. 445-458, 2019.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[7] Cisco, "Joy - A package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring." [Online]. Available: https://github.com/cisco/joy

[8] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[9] R. T. Elaraby, N. M. Abdel Aziem, M. A. Sobh, and A. M. Bahaa-Eldin, "Encrypted network traffic classification based on machine learning," Ain Shams Engineering Journal, vol. 15, 2024.

[10] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time-based features," in Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), 2017, pp. 253-262.