Classification of Video Stream based packets on various Deep Learning and ML approaches

Navnit Kumar
kumar.navnit4175@gmail.com
Concordia University
Montreal, Quebec, Canada

Papry Barua
paprybarua91@gmail.com
Concordia University
Montreal, Quebec, Canada

Abstract

Network packet classification is a critical process in modern network security and management systems, enabling tasks such as intrusion detection, quality of service (QoS) enforcement, and traffic monitoring. With the advent of modern video streaming applications, network traffic patterns have become increasingly complex, requiring robust methods for efficient classification and management. This study explores various approaches to classifying network packets using machine learning techniques and deep packet inspection based deep learning approaches. By leveraging statistical features of packet headers and payloads, the proposed method demonstrates high accuracy in distinguishing between various video streaming applications. Additionally, it incorporates real-time processing capabilities to ensure minimal impact on network performance. Experimental evaluations on 5G traffic dataset [3] show that our approach outperforms traditional rule-based systems and achieves state-of-the-art results in accuracy, precision, and recall. The findings underscore the potential of advanced classification techniques in enhancing network security and efficiency. These findings underscore the potential of advanced machine learning techniques in revolutionizing network traffic management and security infrastructure.

CCS Concepts

• **Network Traffic Classification (NTC)**; • **CNN**; • **LSTM**; • **Decision Tree**; • **Random Forest**;

ACM Reference Format:
Navnit Kumar and Papry Barua. 2024. Classification of Video Stream based packets on various Deep Learning and ML approaches. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The explosive growth of video streaming services has revolutionized content consumption, driving a significant demand for efficient and reliable video delivery over modern networks. As video traffic becomes increasingly complex, accurate classification of video stream packets has become crucial for optimizing network performance, maintaining Quality of Service (QoS), and enhancing user experience. Proper classification facilitates effective bandwidth allocation, congestion control, and dynamic adaptation to varying network conditions. Emerging technologies like Machine Learning (ML) and Deep Learning (DL) provide powerful tools to address the challenges of video stream packet classification. Unlike traditional rule-based methods, which often fail to cope with the diverse and ever-changing characteristics of video traffic—marked by high dimensionality, variable encoding standards, and fluctuating traffic patterns—ML and DL approaches can automatically extract and generalize patterns from raw data, making them particularly suitable for this domain.

Internet service providers (ISPs) play a central role in ensuring optimal QoS and rely on Network Traffic Management (NTM) as a vital mechanism to analyze and regulate traffic behavior. This study focuses specifically on the classification component of NTM. Network Traffic Classification (NTC) can be divided into two primary types: fine-grained classification, also known as conventional NTC, and class-of-service (CoS) classification, often referred to as heap classification. ISPs employ both approaches to optimize IoT networks for improved QoS. Conventional NTC, or multi-class classification, is instrumental in tasks such as resource allocation, monitoring service level agreements, and enhancing network security. However, conventional NTC is resource-intensive and may introduce delays, making it less suitable for real-time scenarios requiring immediate network tuning.

To address such situations, CoS classification is used, enabling quick identification of traffic categories. This method groups services by type (e.g., video, chat, or peer-to-peer), providing sufficient information for immediate resource scheduling and network optimization. For example, while conventional NTC might distinguish between specific services like YouTube, Facebook, or DNS traffic, a CoS classifier identifies broader categories like "video" or "chat." This distinction is especially valuable in dynamic IoT networks, where rapid adaptation is often necessary. CoS classification also plays a critical role in mechanisms such as User Equipment Route Selection Policy (URSP), enabling faster and more efficient traffic management.

By leveraging both conventional and CoS classifiers, ISPs can enhance traffic management processes and deliver better QoS across diverse applications and services. These classifiers offer complementary benefits, with fine-grained classification providing detailed insights for strategic planning and CoS classification supporting real-time network adjustments. In light of these needs, developing robust classifiers remains essential for meeting the demands of modern network environments and evolving user expectations.

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The main contribution of the Paper are

(1) We propose a data pre-processing technique that incorporates domain-specific information to enhance feature differentiation and improve classification accuracy.
(2) We introduce a window-based approach for partitioning time-series data, enabling its transformation into image representations suitable for Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for classification tasks..
(3) We evaluate the performance of various LSTM-based architectures in combination with the transformed data to assess their efficacy in achieving accurate classification results.
(4) We present the results of applying well-known machine learning algorithms on the transformed data, highlighting the most relevant features for optimal classification performance.

2 Related Works

Recent studies have extensively explored various machine learning (ML) techniques to address network traffic classification (NTC) challenges [4] [9][12]. Using the Cambridge University dataset [8], researchers(Yuan & Wang [11]) achieved accuracy levels ranging between 60% and 90% with a decision tree algorithm. In (Liu et al., [?]), the SVM-based STICK mechanism classified 28 different applications with an overall accuracy of 85.98%, achieving up to 99% accuracy for specific applications. Another study (Zhou et al., [13]) utilized a multilayer perceptron (MLP) with three hidden layers to classify 10 class-of-service (CoS) categories, reporting an accuracy of over 96%. Their proposed deep neural network (DNN) model classified traffic from over 200 mobile applications with 93.5% accuracy, leveraging features such as packet size, destination address, protocol, port, and TTL, and employed an eight-layer architecture. Additionally, adaptive learning techniques designed for mobile network traffic have been proposed (Liu et al., [5]), emphasizing reduced classification error rates and faster processing times. These earlier works primarily focused on enhancing classifier accuracy. However, they often targeted limited application sets or classes, with only a few achieving more than 90% accuracy. Meanwhile, Mondal et al. [10] demonstrate the application of Decision Trees (DT) and Random Forest (RF) in Software-Defined Networking (SDN), achieving impressive accuracy.

On the deep learning side recent advancements [1] highlight the effectiveness of deep learning in NTC. Widely adopted architectures include convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, along with hybrid structures like CNN-LSTM.These models extract spatial, temporal, and spatio-temporal features for traffic classification tasks. Lopez-Martin et al. [6] were among the first to apply deep learning for NTC. They utilized CNN, LSTM, and CNN-LSTM architectures to classify traffic from redIRIS, a Spanish academic and research network. Their dataset comprised six features: source port, destination port, packet direction, payload length, inter-arrival time, and window size, with 266,160 network flows. Among these, the CNN-LSTM model achieved the highest accuracy at 96.32%. Similarly, Tong et al. [25] implemented a CNN-based classifier on 20,000 data flows and 1,400 features, achieving an F1-score of 99.34% while classifying five applications. G. Aceto et al.[1] explored various deep learning models for classifying encrypted mobile traffic, treating each byte of a packet as a feature and using zero-padding for normalization. Their comparison of stacked autoencoders (SAE), 2D-CNN, 1D-CNN, LSTM, and hybrid 2D-CNN-LSTM showed 1D-CNN as the best performer, achieving 96.50% accuracy.

While most classifiers focus on application-level features, some recent works, such as [2] rely on radio-level features to reduce dependency on domain-specific assumptions. This study generated its dataset and evaluated CNN and gated recurrent units (GRU) for a multiclass classification problem, achieving 91% accuracy. Another work [15] introduced a multi-scale feature attention-based CNN to analyze specific patterns in network packets per flow, combining initial byte-based raw features with n-gram-based feature maps. This approach achieved 94% accuracy on one dataset and 98.8% on another. A non-linear ensemble method for NTC [16] leveraged well-known algorithms such as SVM, decision trees (DT), random forests (RF), multi-layer perceptron (MLP), AdaBoost, K-nearest neighbors (KNN), LightGBM, CatBoost, and XGBoost. Their ensemble of DT, RF, AdaBoost, and XGBoost achieved 91% accuracy on one of the datasets also used in this study.

2.1 Dataset

The dataset utilized in this study [3] comprises 5G traffic data obtained directly from a leading mobile network operator in South Korea. Traffic measurements were conducted using a Samsung Galaxy A90 5G smartphone, equipped with a Qualcomm Snapdragon X50 5G modem. Packet data was captured using PCAPdroid, a packet-sniffing application installed via Google Play. Traffic was recorded sequentially for individual applications on two stationary devices (only one device was used for non-interactive services), ensuring no background traffic interference.

This dataset encompasses a wide range of traffic types, as detailed in the following table. It includes resource-intensive video traffic, which significantly influences 5G network design and provisioning. Background traffic was excluded to isolate and analyze the distinct characteristics of each traffic type. The video streaming data were captured while using Netflix, a popular over-the-top (OTT) service, on mobile devices. Live streaming traffic was measured while accessing YouTube Live and South Korea's prominent live streaming platforms, such as Naver NOW and Afreeca TV.

Video conferencing traffic was collected during live meetings conducted on widely used platforms like Zoom, Microsoft Teams, and Google Meet. The dataset also includes two types of metaverse traffic: "Zepeto," recorded during 15 hours of interaction in the "Camping" environment, and "Roblox," gathered over 25 hours of gameplay in the "Collect All Pets" mode using an auto-clicker.

Additionally, two types of mobile gaming traffic were recorded. The first category, cloud gaming, involves running games on remote servers and streaming them to user devices, while the second includes conventional mobile games connected to the Internet.

The data collection spanned from May to October 2022, resulting in 328 hours of recorded traffic. The dataset is provided in CSV format as a time-series dataset with timestamps and packet header details. It also includes source and destination addresses, enabling further application-specific traffic analysis.

Type	Application	Protocol	Size
LiveStreaming	YouTube Live	GQUIC	0.73 GB
	AfreecaTV	TCP	4.06GB
	Naver Now	TCP	12.48GB
Stored Streaming	YouTube	QUIC	1.12GB
	Netflix	TCP	0.74GB
Video Conferencing	Zoom	UDP	3.26GB
	MS Teams	UDP	3.71Gb
	Google Meet	UDP	4.41 GB
Metaverse	Zepeto	TCP	0.16GB
	Roblox	RakNet	0.11GB
OnlineGame	Teamfight Tactics	UDP	0.24 GB
	Battleground	UDP	0.38GB
Game Streaming	Geforce Now	UDP	7.05GB
	KT GameBox	UDP	4.36GB

Table 1: Dataset summary

3 Methodology

Our study commenced with a significant focus on effective data processing, which formed the backbone of the subsequent analytical phases. Given the sheer volume and complexity of the data involved, we encountered notable challenges in handling and processing it efficiently. These difficulties came from both the large volume of data and the computational capacity required for its transformation into a format suitable for model training and evaluation. To address these challenges, we leveraged advanced high-performance computing facilities and state-of-the-art tools to expedite the image conversion and dataset tranformation.

The initial phase of our work was devoted to data pre-processing, this step consumed the largest portion of time in this project. This phase involved cleaning, normalizing, and structuring the data to ensure its quality and suitability for training machine learning models and deep learning models.

Simultaneously, after the pre-processing stage, we began training our initial machine learning (ML) models. These preliminary models served as a foundation for understanding the data patterns and allowed us to iteratively refine both the data and the model architectures. In the later stages, we transitioned to more sophisticated techniques by implementing Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) based deep learning models. These advanced models were specifically chosen for their capability to capture both spatial and temporal features effectively, which are intrinsic to the nature of our dataset.

The evaluation of these CNN and LSTM models was conducted systematically, leveraging the insights gained from the earlier ML experiments. This iterative approach not only ensured that the deep learning models were well-optimized but also provided a comprehensive understanding of how different methodologies performed on the same dataset. Through this process, we were able to overcome the initial challenges of data processing and achieve meaningful results in our study. In depth analysis of each step is provided in below subsections.

No	Time	Source	Destination	Protocol	Length	Info

Table 2: Dataset columns provided

Source	Destination	Length	sport	dport
protocol len	P_CLASSIC-STUN		P_CLTP	P_DB-LSP
P_DNS	P_DCP-AF	P_HTTP/JSO	P_SSL	P_TLSv1
P_DTLSv1.2	P_GQUIC	P_HCrt	P_HTTP N	P_MANOLITO
P_Pathport	P_QUIC	P_R-GOOSE	P_RTCP	P_RakNet
P_SSLv2	P_STUN	P_TCP	P_TFTP	P_THRIFT
P_TLSv1.2	P_TLSv1.3	P_UDP		

Table 3: Dataset features extracted

3.1 Data Pre-processing

During the data analysis phase, we identified that the dataset contained a total of 48 unique protocols, each representing different types of network traffic. To facilitate meaningful analysis, we began by working with 7 core columns that effectively described the data, as outlined in the 2. One of the initial steps in our pre-processing pipeline involved parsing the info column, which contained valuable but unstructured data, into more specific features. From this, we extracted additional columns such as sport (source port), dport (destination port), and payload length, all of which were populated with the relevant information parsed from the info column.

We encountered a significant challenge in the dataset regarding the incomplete representation of protocol-related fields, specifically the sport, dport, and payload length attributes. Many rows lacked values for these fields, posing a limitation for effective feature extraction and separability based on protocol information. To address this issue, we adopted a systematic approach to fill in the missing values. For protocols where the dport field was absent, we conducted a detailed review of commonly used destination port values associated with those protocols through authoritative web resources. This mapping allowed us to assign appropriate values to the dport field, ensuring consistency and adherence to protocol specifications. In contrast, the sport field, being application-defined, did not have standard port values. To resolve this, we assigned unique port numbers within the range of 1024–65535 to the missing entries, carefully ensuring that these assignments did not conflict with existing values in the dataset. This approach maintained the integrity of the dataset while enhancing its utility for downstream classification tasks. By augmenting the dataset in this manner, we preserved its accuracy and relevance, enabling a more effective analysis of protocol-specific behaviors. These enhancements also contributed to improved feature distinctiveness, facilitating better classification performance in subsequent stages of the research.

Additionally, the protocol column, which contained categorical protocol names, was transformed into a numerical format using the one-hot encoding technique. This transformation enabled us to represent the protocol information in a way that could be efficiently utilized by machine learning models, eliminating the potential for confusion between protocols with similar names or numeric representations.

As a result of these pre-processing steps, we generated a total of 35 features that were deemed essential for further analysis and model training. These features provided a comprehensive and structured view of the network traffic, allowing us to effectively utilize machine learning models to uncover patterns and make predictions.

ML Models. In the context of our classification task, we employed Random Forest and Decision Tree algorithms as the primary machine learning models for network traffic classification. A key aspect of our data pre-processing involved the transformation of IP addresses into integer values. This transformation was necessary to convert "." separated numbers into a numerical format suitable for machine learning algorithms. Since IP addresses, in their raw form, are textual, encoding them as integers allowed us to preserve their essential information while making the dataset more tractable for the classifiers.

We identified that certain features, such as Time and indexes, were not directly contributing to the classification task and could introduce noise into the model. These non-informative features were removed to reduce dimensionality, ensuring that the models focused on the most relevant attributes of the data.

After these pre-processing steps, we ultimately arrived at a dataset containing 35 3 carefully selected features, which were then used to train our machine learning models. These features were chosen based on their relevance to the network traffic patterns and their ability to distinguish between different classes of traffic. This well-curated feature set provided the foundation for the performance of the Random Forest and Decision Tree classifiers, facilitating accurate and efficient classification of network traffic.

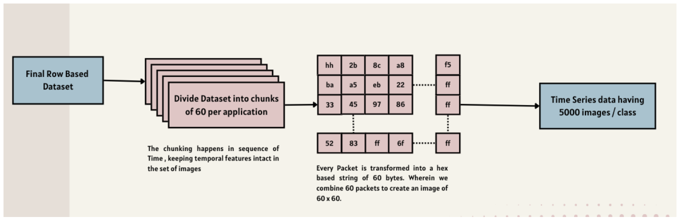
Final Row Based Dataset
Divide Dataset into chunks of 60 per application
The chunking happens in sequence of Time , keeping temporal features intact in the set of images
hh 2b 8c a8 f5
ba a5 eb 22 ff
33 45 97 86 ff
52 83 ff 6f ff
Every Packet is transformed into a hex based string of 60 bytes. Wherein we combine 60 packets to create an image of 60 x 60.
Time Series data having 5000 images / class

Figure 1: Pre-processing Packets to Images

Deep Learning Models. In our approach for applying Convolutional Neural Networks (CNN) to network packet data, we drew inspiration from techniques used in previous studies, such as those outlined in [7]. The core idea behind leveraging CNNs for packet classification is to treat packet data in a way that aligns with the principles of image recognition, as CNNs combined with LSTM have shown remarkable success in processing spatially structured data like images.

To begin, we converted the raw packet data into hexadecimal (hex) representation as shown in 1. The hex codes of network packets, being more compact and structured, provided a way to transform the data into a form of grayscale images. However, it became clear that the hex code of a single packet, might not be enough to create a image big enough which can represent the features of a traffic. In order to address this limitation, we took the approach of chunking the packet data into groups, allowing us to work with a

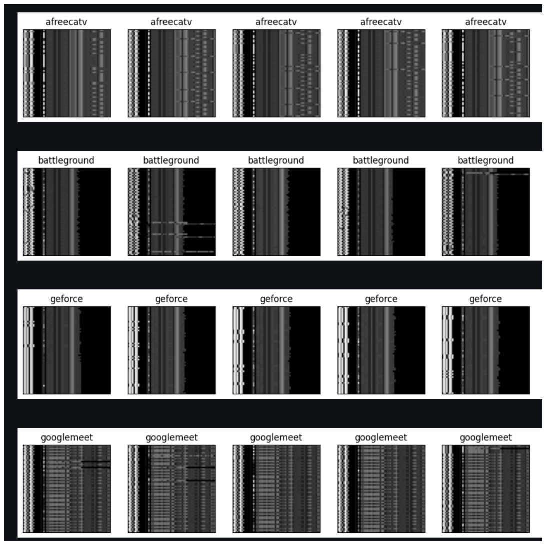

Figure 2: Image generated for afreecatv, battleground, geforce and googlemeet

collection of related packets rather than isolated individual packets. The group numbers were assigned sequentially in order of timing of packets, such that temporal relationship between packets are not lost.

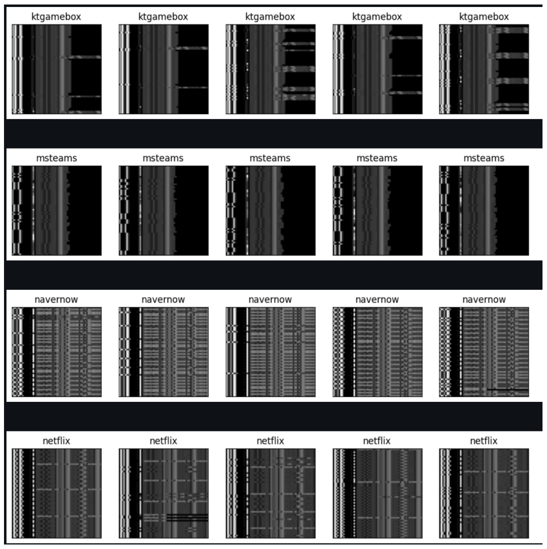

Figure 3: Image generated for ktgamebox, msteams, navernow and netflix

Next, we needed to determine an appropriate size for the input data that it is good enough to accommodate all the packets. To this end, we conducted an analysis of the dataset to compute the mean packet length across the entire collection of network traffic. Based on this statistical analysis, we selected a fixed size of 60 x 60 bytes for the packet transformation. This size was chosen because it effectively captured enough information from the packet data to allow the CNN based models to learn useful patterns, while also being consistent with the overall structure of the network traffic.

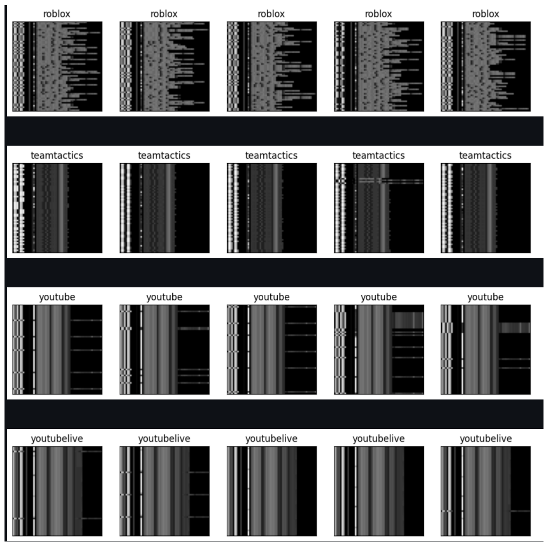

Figure 4: Image generated for roblox, teamtactics, youtube and youtubelive

Given that not all packets in the dataset had payloads of exactly 60 bytes, we had to ensure that every input to the CNN was uniform in size. For packets with total length smaller than 60 bytes, we applied zero-padding (0x00) to the remaining space, ensuring that the packet was extended to the required 60-byte length. This padding allowed us to maintain a consistent input shape for the image, while not introducing any new meaningful data. On the other hand, for packets that exceeded 60 bytes, we chose to truncate or ignore the excess data, as our approach was designed to focus on packets with a standard length of 60 bytes, which we considered the optimal representation for our task.

The final transformation resulted in a set of 60 x 60 matrices representing the network packets, where each matrix served as an image-like input to the CNN as shown in 23 4 5. This innovative approach of treating packet data as 2D matrices allowed the CNN to leverage its strength in identifying spatial patterns, such as correlations between packet byte values and traffic characteristics, in order to improve classification accuracy.

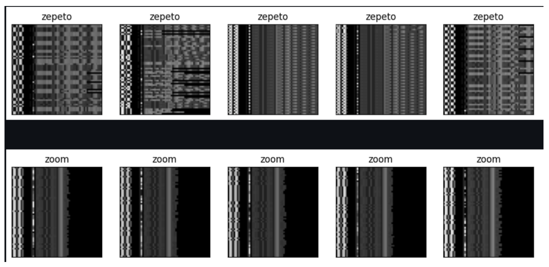

Figure 5: Image generated for zepeto and zoom

3.2 Model Implementation and Training

For the purpose of evaluating the performance of our machine learning (ML) and deep learning (DL) models, we adopted a three-way data split approach, partitioning the dataset into training, testing, and validation sets. Specifically, 60% of the data was allocated for training the models, while 20% was reserved for testing and 20% for validation. This partitioning scheme is commonly utilized in many machine learning workflows to ensure that models are trained on a sufficiently large portion of the data, while also maintaining a distinct and unbiased dataset for both validation and testing.

While it is recognized that in some machine learning paradigms, such as those involving highly structured or sequential data, this specific data splitting ratio might not always be the most optimal, it is a widely accepted practice in many standard ML methodologies. This conventional approach is particularly useful in ensuring that both the validation and testing phases are conducted on independent, non-overlapping subsets of data, thereby mitigating the risk of over fitting and providing a more reliable evaluation of the model's generalization ability.

Further training details per models, as well as its implications on model evaluation, are discussed in the subsequent sections.

3.2.1 ***ML Models.*** Decision Trees (DT) are a widely used supervised learning technique due to their simplicity and strong performance in classification tasks. The DT algorithm constructs a model in the form of a tree, where each internal node represents a decision based on a feature, and each leaf node corresponds to a predicted class label. DTs are favored for their interpret-ability, ease of use, and ability to handle both numerical and categorical data. The primary goal of the DT algorithm is to split the dataset into subsets based on feature values, ultimately leading to a final prediction rule. Despite their effectiveness, Decision Trees can suffer from over fitting, especially when the model becomes too complex or the dataset includes noise.

The most popular Decision Tree algorithms include CART (Classification and Regression Trees) (Burrows et al., Citation1995), ID3 (Iterative Dichotomiser 3) (Quinlan, Citation1986), and C4.5 (Karatsiolis & Schizas, Citation2012). Each algorithm has its own method for constructing the tree, with C4.5, in particular, excelling in handling statistical data due to its use of entropy-based splitting criteria and pruning techniques to reduce model complexity. Given its robustness, we selected the C4.5 algorithm with a random state as 10 for our study, ensuring reproducibility and reliable results in training the model.

Although the Decision Tree algorithm performed well in many cases, we observed that it tends to overfit the training data. Overfitting occurs when the model learns not only the underlying patterns in the data but also the noise or anomalies, which can negatively impact the model's ability to generalize to new data.

To mitigate the overfitting issue, we employed Random Forest (RF), an ensemble learning method that builds multiple Decision Trees and combines their predictions to improve classification performance. By training a collection of trees using random subsets of the training data and feature set, RF is less prone to overfitting compared to a single DT, as it reduces the variance in predictions. Each tree in the forest is trained independently on different data points, and the final prediction is made by averaging the outputs of all trees, resulting in better generalization.

To prevent further overfitting in the Random Forest model, we set the maximum depth and random state parameters to fixed values of 10 and 15, respectively. These choices were made based on experimentation, aiming to balance model complexity with predictive accuracy. By constraining the depth of the trees and controlling the randomness, we ensured that the RF model would remain robust while minimizing the risk of overfitting.

In conclusion, Decision Trees provide a simple yet effective classification approach, but they are susceptible to overfitting, especially in complex datasets. Random Forest, by leveraging multiple trees, improves classification performance and generalization, though certain data types may still require more specific attention. Future work could focus on enhancing the performance of these classes through advanced techniques, such as more targeted feature extraction or

3.2.2 ***Deep Learning Models.*** Time-distributed deep learning plays a crucial role in extracting comprehensive temporal features from data, especially in network traffic analysis. This approach facilitates the extraction of both short-term and long-term temporal dependencies by utilizing specialized models such as CNNs and LSTMs, each contributing uniquely to the feature extraction process. The spatial features, derived through Convolutional Neural Networks (CNNs), are designed to capture the spatial patterns in the data, often referred to as intra-temporal features. These features are particularly important in network traffic as they reveal local patterns and spatial dependencies within the data.

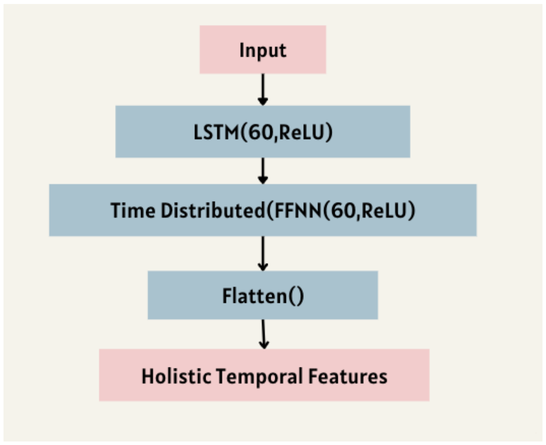

Figure 6: Implementation details of Model 2. 60 units of LSTM are employed in the first layer with ReLU activation. The initial layer of LSTM extracts the inter-temporal features, which are fed to the time-distributed FFNN of 60 units. The flattened final output consists of inter- and pseudo-temporal features

Long Short-Term Memory (LSTM) networks are utilized alongside spatial features to capture the temporal patterns within the data, often referred to as inter-temporal features. LSTMs are highly effective in identifying sequential dependencies, making them well-suited for analyzing the evolution and changes in network traffic over extended sequences. By integrating Convolutional Neural Networks (CNNs) with LSTM architectures, the framework benefits from the simultaneous extraction of both spatial and temporal characteristics. This synergy is essential for comprehending the complex dynamics of network traffic flows. The temporal features derived through this combination are termed pseudo-temporal features, as they encapsulate temporal dependencies arising from both intra-temporal and inter-temporal information. Deep learning models

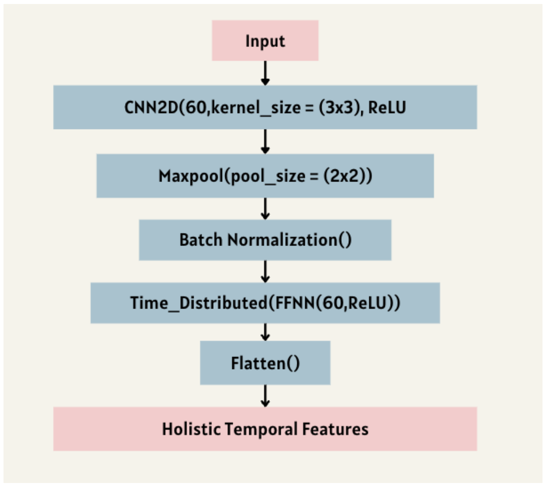
Input
CNN2D(60,kernel_size = (3x3), ReLU
Maxpool(pool_size = (2x2))
Batch Normalization()
Time_Distributed(FFNN(60,ReLU))
Flatten()
Holistic Temporal Features

Figure 7: Implementation details of Model 1. 60 units of 2D CNN are used with a 3×3 kernel window. 2 × 2 maxpool is introduced between CNN2D and batch normalization layers. The output of batch normalization consists of intra- or spatio-temporal features and is fed to 60 units of time-distributed FFNN. The extracted features consist of intra- and pseudo-temporal features.

are often regarded as "black boxes" due to their inherent complexity, which makes interpreting or explaining the exact features they learn a challenging task. Nevertheless, we aim to understand and generalize the different types of features these models capture. To facilitate this, we introduce a time-distributed layer positioned after the primary deep learning layers. This layer is specifically designed to enhance feature extraction by incorporating a feed-forward neural network (FFNN) within a time-distributed framework. The time-distributed configuration enables the FFNN to process each time step of the input sequence independently, thereby supporting parallel feature extraction for individual temporal segments. This parallel approach is particularly advantageous for sequential data, as it allows the model to uncover intricate temporal patterns from each segment effectively.

In our approach, we define three specific models for extracting holistic temporal features, all incorporating a time-distributed deep learning layer. These models include:

(1) **Model 1: CNN-TD(FFNN)** – This model combines a Convolutional Neural Network (CNN) for spatial feature extraction with a time-distributed feed-forward neural network (FFNN) for temporal feature extraction as shown in 6.

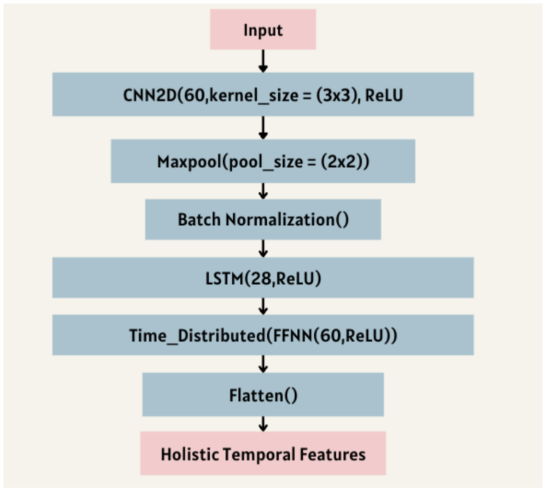
Input
CNN2D(60,kernel_size = (3x3), ReLU
Maxpool(pool_size = (2x2))
Batch Normalization()
LSTM(28,ReLU)
Time_Distributed(FFNN(60,ReLU))
Flatten()
Holistic Temporal Features

Figure 8: Implementation details of Model 3. Model 3 uses CNN2D and LSTM along with time-distribution FFNN. We use CNN2D as the first layer to utilize the advantage of high-dimensional data and to reduce the data dimension for future layers. It extracts the spatio-temporal features at the beginning to improve performance. The spatio-temporal data extracted by CNN is fed to LSTM of 60 units. LSTM helps to extract the interflow features. Finally, the time-distributed FFNN extracts the pseudo-temporal features. The flattened data after time-distributed FFNN is nothing but holistic-temporal features.

(2) **Model 2: LSTM-TD(FFNN)** – This model employs a Long Short-Term Memory (LSTM) network to capture temporal dependencies, followed by a time-distributed FFNN for enhanced temporal feature learning as shown in 7.
(3) **Model 3: CNN-LSTM-TD(FFNN)** – This model combines both CNN and LSTM to extract spatial and temporal features in parallel, with a time-distributed FFNN to further refine the temporal feature learning. By incorporating time-distributed deep learning techniques, we expect to improve the model's ability to generalize over time, leading to more robust and efficient traffic classification in various network settings as shown in **??**

In summary, the combination of CNNs, LSTMs, and time-distributed layers enables the model to extract rich spatial and temporal features, providing a more comprehensive understanding of network traffic flows. The empirical results will demonstrate the advantages of using time-distributed deep learning in improving the classification accuracy, especially in capturing complex temporal dependencies that are vital for real-world applications such as network traffic analysis.

4 Evaluation

The evaluation of both models was conducted using the validation split derived from the same dataset. For the machine learning models, the evaluation process was straightforward: following the training phase, the models were tested on both the test and validation splits to assess performance.

In contrast, the deep learning models were subjected to additional testing by randomizing the time series data. This approach was designed to evaluate the models' ability to effectively learn and interpret features within the dataset.

All experiments were performed on an Apple M4 Pro Notebook equipped with a 16-core graphical processing unit (GPU). The required software modules were implemented in Python, utilizing libraries such as TensorFlow, Keras, and sklearn to build and evaluate the models.

Model	F1	Precision	Recall	Accuracy
Decision Tree	0.99	0.99	0.99	0.98
Random Forest	0.99	0.99	0.99	0.99

Table 4: ML-Model Based Results

Model	F1	Precision	Recall	Accuracy
CNN-TD(FFNN)	1.00	0.9982	1.00	1.00
LSTM-TD(FFNN)	1.00	0.9968	1.00	1.00
CNN-LSTM-TD(FFNN)	1.00	0.9962	1.00	1.00

Table 5: CNN & LSTM Based Results

4.1 Performance Metrics

We compute the performance metrics using the predicted classes and the ground truth labels. Specifically, we quantify the performance of the classification models using Accuracy, Precision, Recall, and F1-score as given by.

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

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

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

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

where false-positive (FP) represents the false detection of the flows when the flow is not present in reality, false-negative (FN) represents no detection when the flows exist in reality, true-negative (TN) is the correct no-detection, and true-positive (TP), on the contrary, represents the correct detection. We also study the total time to finish the training. The Accuracy metric provides how many test samples are classified correctly. Precision explains the number of test samples correctly predicted in the cases where the samples were positive. Recall indicates how many actual positive cases are predicted correctly using our model. F1-score is the harmonic mean of Precision and Recall. We have a higher F1-score when a model achieves balanced Precision and Recall. Therefore, a model with a higher F1-score is always better in the classification. In addition, we evaluate the total training time from the Keras application interface.

5 Results

The evaluation phase yielded various notable results, which are detailed below:

(1) Traditional machine learning models such as Decision Tree and Random Forest were implemented as baseline methods to evaluate their effectiveness in traffic classification. These models, while relatively simple and computationally efficient, primarily rely on static features and fail to analyze temporal relationships inherent in time-series data. We can analyse that DT overall demonstrates 98% of accuracy, however, it has the tendency to over-fit. In such cases, RF imparts marginally better accuracy of 99%, which restrain the problem of over-fitting. Although DT and RF have such higher overall accuracy, they achchieve satisfactory accuracy and f1-score as seen in 4

(2) The hybrid CNN + LSTM model significantly outperformed traditional methods by leveraging its dual-feature extraction capabilities: Intra-packet Features: The CNN component efficiently captured spatial features within individual packets, such as distribution patterns and static attributes Inter-packet Features: The LSTM component excelled in analyzing temporal dependencies across packets, such as sequential trends and time-based variations. This combination made the CNN + LSTM model particularly well-suited for handling multi-class time-series traffic data, enabling it to achieve superior classification results and address challenges that traditional models could not overcome as shown in 5.

6 Future Work

While the approaches employed in this study have demonstrated their effectiveness, certain limitations have been observed, particularly the tendency of models to overfit due to the use of entries from the same dataset for both training and testing. This is reflected in the consistently high accuracy across all approaches. To address this, future work could involve testing the models on datasets with randomized traffic to provide a more robust validation of their accuracy.

For the ML models, although they have been implemented with careful feature engineering, their performance can potentially be enhanced through advanced hyperparameter tuning techniques.

In the CNN-based models, the introduction of padding in images has resulted in computations involving meaningless elements (e.g., zero-padding). To mitigate this, Sparse CNN techniques could be explored to prune irrelevant weights in the convolutional layers, thereby improving training efficiency. Furthermore, advanced preprocessing methods, such as utilizing window-based summarization of flows, could be investigated to extract more meaningful features from the data.

Additionally, the large size of the dataset required splitting it into smaller chunks for manageable processing. Future work could focus on generalizing the training process to accommodate larger datasets more effectively and efficiently.

7 Conclusion

This study provided a robust solution for classifying time-series video streaming traffic using a hybrid CNN + LSTM model, which successfully combined spatial and temporal feature extraction. By transforming raw packet data into structured 60x60 images, the approach utilized CNNs to identify spatial patterns, such as packet size distributions, and LSTMs to capture sequential dependencies like packet arrival times. This dual approach enabled the model to accurately classify traffic types, overcoming limitations of traditional machine learning methods. Models such as Decision Trees and Random Forests, while computationally efficient, failed to handle the inherent temporal complexities in the data, underscoring the superiority of the deep learning approach. The results of this study demonstrated the potential for real-time traffic classification in addressing critical issues faced by Internet Service Providers (ISPs),such as fairness among different traffic types and ensuring improved Quality of Service (QoS). The CNN + LSTM model showed high accuracy across multiple traffic classes, proving its effectiveness in understanding the nuanced dynamics of video streaming traffic. This success highlights the growing importance of deep learning methodologies in modern network traffic analysis.

References

[1] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. 2019. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE transactions on network and service management* 16, 2 (2019), 445–458.

[2] Miguel Camelo, Paola Soto, and Steven Latré. 2021. A general approach for traffic classification in wireless networks using deep learning. *IEEE Transactions on Network and Service Management* 19, 4 (2021), 5044–5063.

[3] Yong-Hoon Choi, Daegyeom Kim, and Myeongjin Ko. 2023. 5G Traffic Datasets. https://doi.org/10.21227/ewhk-n061

[4] K Chokkanathan and S Koteeswaran. 2018. A study on flow based classification models using machine learning techniques. *International Journal of Intelligent Systems Technologies and Applications* 17, 4 (2018), 467–482.

[5] Zhen Liu, Nathalie Japkowicz, Ruoyu Wang, and Deyu Tang. 2019. Adaptive learning on mobile network traffic data. *Connection Science* 31, 2 (2019), 185–214.

[6] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. 2017. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE access* 5 (2017), 18042–18050.

[7] Yoga Suhas Kuruba Manjunath, Sihao Zhao, Xiao-Ping Zhang, and Lian Zhao. 2024. Time-Distributed Feature Learning for Internet of Things Network Traffic Classification. *IEEE Transactions on Network and Service Management* (2024).

[8] Andrew Moore, Denis Zuev, and Michael Crogan. 2013. *Discriminators for use in flow-based classification.* Technical Report.

[9] Tiago Prado Oliveira, Jamil Salem Barbar, and Alexsandro Santos Soares. 2016. Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *International Journal of Big Data Intelligence* 3, 1 (2016), 28–37.

[10] Emmanuele Benedetto Yao Shen Pritom Kumar Mondal, Lizeth P. Aguirre Sanchez and Minyi Guo. 2021. A dynamic network traffic classifier using supervised ML for a Docker-based SDN network. *Connection Science* 33, 3 (2021), 693–718. https://doi.org/10.1080/09540091.2020.1870437 arXiv:https://doi.org/10.1080/09540091.2020.1870437

[11] Zhengwu Yuan and Chaozheng Wang. 2016. An improved network traffic classification algorithm based on Hadoop decision tree. In *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*. IEEE, 53–56.

[12] Jun Zhang, Chao Chen, Yang Xiang, and Wanlei Zhou. 2012. Semi–supervised and compound classification of network traffic. *International Journal of Security and Networks* 7, 4 (2012), 252–261.

[13] Wengang Zhou, Leiting Dong, Lubomir Bic, Mingtian Zhou, and Leiting Chen. 2011. Internet traffic classification using feed-forward neural network. In *2011 International conference on computational problem-solving (ICCP)*. IEEE, 641–646.