RIGA TECHNICAL UNIVERSITY

Tianhua Chen

FINE-GRAINED MONITORING AND CLASSIFICATION OF INTERACTIVE MEDIA NETWORK TRAFFIC

Doctoral Thesis

RTU Press
Riga 2026

RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science, Information Technology and Energy
Institute of Photonics, Electronics and Telecommunications

Tianhua Chen
Doctoral Student of the Study Programme "Telecommunications"

FINE-GRAINED MONITORING AND CLASSIFICATION OF INTERACTIVE MEDIA NETWORK TRAFFIC

Doctoral Thesis

Scientific supervisors:

Associate Professor Dr. sc. ing.
ELANS GRABS

Associate Professor Dr. sc. ing.
ALEKSANDRS IPATOVS

Co-advisor:
Professor PhD
CANO BAÑOS, MARÍA DOLORES

RTU Press
Riga 2026

Chen, T. Fine-Grained Monitoring and Classification of Interactive Media Network Traffic. Doctoral Thesis. Riga: RTU Press, 2026. 143 p.

Published in accordance with the decision of the Promotion Council "P-08" of 12 November 2025, Minutes No. 43.

This Thesis has been supported by the EU Recovery and Resilience Facility within the Project No. 5.2.1.1.i.0/2/24/I/CFLA/003 "Implementation of consolidation and management changes at Riga Technical University, Liepaja University, Rezekne Academy of Technology, Latvian Maritime Academy and Liepaja Maritime College for the progress towards excellence in higher education, science and innovation" (ID 1072).

Cover picture from ChatGPT.

DOCTORAL THESIS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF SCIENCE

To be granted the scientific degree of Doctor of Science (PhD), the present Doctoral Thesis has been submitted for defence at the open meeting of RTU Promotion Council on 13 February 2026 at 11.00 AM at the Faculty of Computer Science, Information Technology and Energy of Riga Technical University, Āzenes Street 12, Room 201.

OFFICIAL REVIEWERS

Professor Dr. sc. ing. Sandis Spolītis
Riga Technical University

Professor Dr. sc. ing. Jaime Lloret
Technical University of Valencia, Spain

Professor Dr. sc. ing. Mario Alberto Montagut Climent
University of Valencia, Spain

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for review to Riga Technical University for promotion to the scientific degree of Doctor of Science (PhD) is my own. I confirm that this Doctoral Thesis has not been submitted to any other university for promotion to a scientific degree.

Tianhua Chen ……. (signature)
Date: ……

The Doctoral Thesis is written in English and consists of four chapters, 16 figures, 23 tables, and 8 appendices. The total length is 143 pages, including appendices. The bibliography contains 157 references.

ABSTRACT

Network Traffic Monitoring and Analysis (NTMA) plays a critical role in assessing network performance and optimizing resource utilization. In today's heterogeneous network environments, interactive media applications consume a substantial portion of network traffic. Emerging services such as cloud gaming, virtual reality (VR)/metaverse, and live streaming with 4K/8K resolution and 60/120 frame rates content are rapidly expanding and significantly increasing traffic volume. In conventional Network Traffic Classification (NTC) tasks, traffic generated by these applications is often broadly categorized as video traffic. These coarse labeling limits precision and leads to underrepresentation in many public Internet traffic datasets. Moreover, even traffic from the same application or service may display different characteristics under varying network conditions in heterogeneous environments.

To address this gap, the doctoral thesis aims to achieve fine-grained monitoring and classification of network traffic generated by interactive media applications across diverse network conditions. The classification is conducted at three granularity levels: time series, flow, and payload. This approach enhances understanding of interactive media traffic patterns and their roles in complex network environments. It also contributes to traffic behavior modeling for NTC tasks and extends to broader NTMA applications. To support this goal, four categories of interactive media applications were selected. Corresponding datasets were created at each granularity level. Five deep learning and ensemble models were developed and evaluated against three state-of-the-art classifiers to assess classification performance.

CONTENTS

LIST OF ABBREVIATIONS

AAC	Advanced Audio Coding
ABR	Adaptive BitRate
AI	Artificial Intelligence
BiRNN	Bi-directional Recurrent Neural Network
CBR	Constant BitRate
CCDF	Complementary Cumulative Distribution Function
CDN	Content Delivery Network
CNN	Convolutional Neural Network
CV	Coefficient of Variation
DASH	Dynamic Adaptive Streaming over HTTP
Db	Daubechies
DBN	Deep Belief Network
DLSS	Deep Learning Super Sampling
DPI	Deep Packet Inspection
DTLS	Datagram Transport Layer Security
EBSNN	Extended Byte Segment Neural Network
ENN	Edited Nearest Neighbors
FPS	Frames Per Second
GCN	Graph Convolutional Network
GI	General Independent
GNN	Graph Neural Network
GOP	Group of Pictures
GRU	Gated Recurrent Unit
GSO	Generic Segmentation Offload
GTP-U	GPRS Tunnelling Protocol User Plane
HLS	HTTP Live Streaming
HTTP	Hyper Text Transfer Protocol
HTTP-FLV	HTTP FLV live stream
IAT	InterArrival Time
ICE	Interactive Connectivity Establishment
IDC	International Data Corporation
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
ISP	Internet Service Providers
KNN	K Nearest Neighbors
LAN	Local Area Network
LSTM	Long Short Term Memory
MAC	Media Access Control
MLP	Multi-Layer Perceptron
MPD	Media Presentation Description
MSE	Mean Squared Error
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NGAP	Next Generation Application Protocol
NTC	Network Traffic Classification
NTMA	Network Traffic Monitoring and Analysis
OTT	Over The Top
PCAP	Packet Capture
PDF	Probability Density Function
QoE	Quality of Experience
QoS	Quality of Service
QUIC	Quick UDP Internet Protocol
RAN	Radio Access Network
RFC	Request for Comments
RLC	Radio Link Control
RNN	Recurrent Neural Network
RTCP	Real Time Control Protocol
RTMP	Real Time Messaging Protocol
RTP	Real-Time Protocol
SMOTE	Synthetic Minority Oversampling TEchnique
SNI	Server Name Indication
SRS	Simple Realtime Server
SRT	Secure Reliable Transport
SSIM	Structural Similarity Index
SSL	Secure Sockets Layer
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TFSN	TLS Flow Sequence Network
TLS	Transport Layer Security
Tor	The onion router
TPR	True Positive Rate
TSO	TCP Segmentation Offload
TURN	Traversal Using Relays around NAT
UHD	Ultra High Definition
UPF	User Plane Function
VBR	Variable BitRate
VDS	Virtual Desktop Streaming
VLC	VideoLAN Client
VLS	Video Live Streaming
VoD	Video on Demand
VPN	Virtual Private Network
WebRTC	Web Real Time Communication
XR	eXtended Reality

LIST OF SYMBOLS

$\boldsymbol{t}$	Vector of packet arrival epoch time
$\boldsymbol{\Delta t}$	Vector of IAT
$\mu_{\Delta t}$	Mean of $\boldsymbol{\Delta t}$
$\sigma_{\Delta t}$	Standard deviation of $\boldsymbol{\Delta t}$
$CV_{\Delta t}$	Coefficient of variation of $\boldsymbol{\Delta t}$
$\{\mathrm{x_i}\}_{\mathrm{i=1}}^{\mathrm{N}}$	A sequence of observed values
s_i	Service time for packet i
$h(x_j)$	Count of observations in j-th histogram bin
$\hat{p}(x_j)$	Normalized PDF
M	Total number of observations
Δx_j	The width of the j-th histogram bin
$\mathbf{W}$	Logarithmically spaced windows vector
$\mathbf{V}$	Variance of aggregated packet counts over time windows vector
λ_w	Arrival rate of the specific sliding window
$\mathbf{S}_w$	Service time vector of all sliding windows
ρ_w	Traffic intensity of the specific sliding window
$\mathbf{W}_q$	Waiting queue time vector of all sliding windows
$\mathbf{T}_w$	Total delay vector of all sliding windows
$\mathbf{J}_w$	Jitter vector of all sliding windows
$\mathbf{L}_q$	Queue length vector of all sliding windows
$\mathbf{B}_w$	Buffer size vector of all sliding windows
$\boldsymbol{p}_{loss}$	Packet loss rate vector of all sliding windows
$\boldsymbol{\mathcal{T}}$	Packet arrival relative time vector
$\mathcal{I}$	Traffic intensity vector expressed by the time series
$\mathcal{T}_{traf}$	Traffic intensity vector corresponding time bins vector
$\boldsymbol{\mathcal{P}}$	Packet arrival payload length vector
$\mathcal{D}$	Dataset
(w, h)	Image size of width and height
$\mathcal{R}$	Raw frames in the PCAP file
B	Batch size
C	Number of classes
D	Input feature size
F	Number of output channels
F_{1D}	Total filter operations in 1D convolutional layers
F_{2D}	Total filter operations in 2D convolutional layers
L	Length of 1D sequence in 1D-CNN/RNN
n_{layers}	Number of RNN/GRU layers
F_{gcn}	Feature dimension per node
H	Number of hidden units in RNN/GRU layers
H_s	Spatial height of 2D input
W	Spatial width of 2D input
$\lvert E \rvert$	Number of edges
N	Number of training samples
M	Number of meta learners
T	Number of estimators

LIST OF TABLES

LIST OF GRAPHS

1. OVERVIEW

1.1. Introduction

According to International Data Corporation (IDC), global Internet users are projected to generate 175 ZB of traffic data by 2025 [1]. Network traffic is a complex collection of interactions, data flows, and endpoints characterized by self-similarity, burstiness, and cyclical patterns. In the face of rapidly increasing data traffic, Network Traffic Monitoring and Analysis (NTMA) plays a crucial role in monitoring network performance and improving resource utilization [2], [3]. Network Traffic Classification (NTC) is one of the most crucial applications in the NTMA. NTC is the process of categorizing network traffic based on network characteristics to identify different applications or services running on the network [4]–[12]. This classification is essential for Internet Service Providers (ISPs) to manage their operations effectively. Categorizing traffic into priority levels helps ensure that critical applications receive adequate bandwidth and processing power. It also aids in anomaly detection and implementing measures to maintain network security and performance. Meanwhile, video-based services have become a major contributor to network traffic data [13]. According to Cisco, video media will account for 82 % of all IP traffic by 2021 [14]. Over-the-top (OTT) applications, such as Netflix and YouTube, dominate this growth, with Sandvine Incorporated reported a 24 % increase in video traffic as of January 2023 [15], [16]. Currently, many prevailing OTT platforms offer 4K/8K Ultra-High-Definition (UHD) resolution content with high frame rates like 60 FPS or 120 FPS, significantly impacting network traffic volume [17]–[24].

Most existing studies on media application traffic monitoring and classification focus primarily on encrypted OTT video applications [25]–[28]. Researchers have collected large volumes of network traffic and constructed several public benchmark datasets, including ISCXVPN2016, ISCXTor2016, MIRAGE-2019, CICDarknet2020, MAppGraph2021, and UTMobileNet2021 [29]–[34]. These applications are labeled based on features including application name, streaming platform, audio and video content, encapsulation, encryption, anonymization methods, transmission protocols, quality of service (QoS) levels, quality of experience (QoE), and other characteristics [35]–[41]. With the use of advanced machine learning and deep learning classifiers, these studies have achieved excellent classification performance at multiple granularity levels, including payload and flow analysis [42]–[86]. Although video-related traffic is present in these datasets, it usually represents only a small portion of the total data. Some proprietary datasets are specifically focused on video traffic, using classification labels derived from metadata such as playback behavior, and video parameters [87]–[94]. However, these datasets still have several limitations. There is often a weak correlation and explainability between the samples and their ground-truth labels, the classification boundaries remain unclear, and the data is frequently mixed with other types of application traffic. This overlap occurs across various dimensions, including network protocols, system architectures, application categories, and traffic distribution patterns, which makes the media application traffic profiles more complex [95], [96].

Additionally, emerging applications such as cloud gaming, social networking, web conferencing, and eXtended Reality (XR)/Metaverse experiences are evolving rapidly [97]–[99]. Most existing works focus on traffic modeling and analysis, while traffic monitoring and classification for emerging applications remain in the early stages of research [100]–[108]. However, these emerging applications are evolving rapidly, making it increasingly difficult to categorize network traffic using traditional methods based on applications or services. The growing convergence and interactivity among applications have led to diminishing distinctions in their traffic characteristics. To gain deeper insights into the traffic patterns of these next-generation applications, fine-grained monitoring and classification of interactive media network traffic have become essential. For OTT providers, it enables them to anticipate trends in interactive media network traffic, optimize network configurations, enhance performance, reduce security risks, and deliver better video services to users.

1.2. Rationale

To achieve fine-grained monitoring and classification of interactive media network traffic, a structured multi-step methodology is required. The first essential step involves comprehensive traffic data collection. To ensure diversity and representativeness, interactive media traffic is categorized into three major application domains: Live Streaming, Cloud Gaming, and Metaverse. The Live Streaming category includes traffic from five globally and regionally popular platforms: YouTube, BiliBili, Twitch, Facebook, and TikTok. The Cloud Gaming category consists of traffic captured from applications such as *Spitlings*, *Tomb Raider*, and *Thumper* [98]. The Metaverse category comprises applications including *VRChat*, *TheLab*, *SolarSystemAR*, *RealityMixer*, *Hellblade*, *DiRT Rally 2.0*, and *Bigscreen Theatre* [108]. All traffic sources are obtained from public and proprietary datasets and exhibit variation in network standards, capture durations, sniffing tools, and device platforms. Therefore, a critical preprocessing step is necessary to normalize and prepare the data for subsequent analysis.

The second step involves determining the appropriate number and structure of traffic categories to support fine-grained classification. Key factors influencing traffic characteristics, particularly for video-based applications, include resolution, bitrate, encoding format, container type, frame rate, streaming mode such as Video on Demand (VoD) or Video Live Streaming (VLS), and streaming protocols. These parameters, along with prior domain knowledge, help define meaningful and distinguishable traffic categories. For encrypted traffic, the classification granularity significantly affects both accuracy and efficiency. While many studies focus on flow-level and payload-level classification, time series-based approaches have become increasingly relevant. These methods utilize spatiotemporal patterns in traffic data without relying on payload inspection, making them well suited for privacy-preserving classification tasks. In the fourth step, datasets are constructed based on the selected classification granularities. The final step involves selecting relevant training features and applying supervised learning algorithms to perform the classification task. Model performance is then evaluated using standard classification metrics to assess the effectiveness of fine-grained monitoring and classification of interactive media network traffic [13], [2], [3].

1.3. The Aim and Theses of the Doctoral Thesis

The objective of this Doctoral Thesis is to achieve fine-grained monitoring and classification of network traffic generated by interactive media applications across diverse network conditions at multi-granularity levels. This study aims to improve the understanding of traffic patterns and the roles of interactive media in today's complex and heterogeneous network environments, while also contributing to traffic behavior analysis in NTC and extending to NTMA tasks. The core findings are summarized as follows:

Thesis 1: Network bandwidth is the primary factor affecting interactive media network traffic volume and distribution. Additionally, video resolution and frame rates also play a direct role in shaping traffic characteristics. Variable bitrate encoding, fluctuations in keyframe bitrate, and the utilization of GOP structures are among the most critical factors affecting traffic behaviors.

Thesis 2: Streaming application protocols can dynamically pair with different transport layer protocols. When WebRTC is combined with UDP, it produces the highest number of packets but with the smallest payload size, where the traffic volume accounts for only 70 % of the corresponding raw video file size, while also showing the lowest latency. When HLS is combined with TCP, it yields the largest total payload volume, up to 1.5 times larger than the raw video file under the same network conditions, and exhibits a high control-plane to data-plane traffic ratio.

Thesis 3: In fine-grained classification tasks, ground truth labels are directly associated with sample characteristics. These labels are derived from traffic-influencing factors and other prior knowledge. All training features and samples are transparent and interpretable. The classification task is conducted at three granularities – time series, flow, and payload – making the approach adaptable to different scenarios.

Thesis 4: The Stacking model achieved over 99 % overall classification accuracy across 86 categories in the Live Streaming dataset, containing more than 200,000 samples. Each category attained an F1-score of at least 0.94 at the flow level. The 1D-CNN model achieved at least 80 % classification accuracy at both the time series and flow levels, outperforming three state-of-the-art deep learning models. At the payload level, the 2D-CNN model reached over 84 % classification accuracy, exceeding other methods by more than 10 %.

The research hypotheses of the Doctoral Thesis:

1. Different interactive media applications generate similar traffic volumes under the same network bandwidth. Variations in raw video parameters result in traffic volume differences of no more than 10 % among all applications. Setting the GOP to 180 frames best reflects the actual traffic burstiness distribution over time.

2. WebRTC, combined with UDP, transmits the smallest payload volume, achieving over 30 % traffic savings compared to the original video size. It also generates at least twice as many packets as other streaming protocols, including HLS, DASH, RTMP, and HTTP-FLV.

3. Labels in fine-grained classification correlate with known traffic generation parameters, allowing for transparent and explainable class mapping. At the payload level, the total number

of interactive media traffic samples exceeds that of the time-series and flow levels by at least 70 %, even without resampling techniques.

4. The 2D-CNN model achieves over 84 % classification accuracy at the payload level, outperforming other baselines by at least 10 %.

1.4. The Tasks of the Doctoral Thesis

1. To collect interactive media application network traffic from multiple sources, including public and proprietary datasets. Traffic was actively captured from various platforms, applications, and services under different network conditions. Bidirectional traffic from both client and server sides was obtained in idealized network environments across multiple streaming protocols.

2. To use the standard video quality testing file, Big Buck Bunny, for client-server push and pull stream tests. To generate video files with varying quality levels using the `FFmpeg` video transcoding tool and examine the relationships between bitrate and frame distribution.

3. To conduct global and local statistical analysis of interactive media network traffic. To employ a sliding time window method to analyze traffic distributions across varying time scales and identify patterns and behaviors.

4. To develop algorithms to generate classification datasets for interactive media applications network traffic at three levels of granularity: time series, flow, and payload. To apply preprocessing to construct the corresponding datasets.

5. To design and evaluate classification models including 1D-CNN, 2D-CNN, RNN, GCN, and stacking. To compare their performance with three state-of-the-art models: Deep Packet, FlowPic, and FS-Net.

1.5. Research Methods

To carry out the tasks outlined in the Doctoral Thesis and analyze the associated problems, `Wireshark`, `Tcpdump`, and `nDPI` tools were employed to collect diverse network traffic. Preprocessing was conducted using `MATLAB`, `NFStream`, and `CICFlowMeter` to generate proprietary datasets, which were also compared with various public benchmark datasets. For mathematical modeling, multiple state-of-the-art machine learning algorithms were applied while using popular Python libraries, including `PyTorch`, `PyTorch Geometric`, `OpenCV`, `SciPy`, `StellarGraph`, `TensorFlow`, `Keras`, `XGBoost`, `scikit-image` and `scikit-learn` for network traffic classification. Additionally, `srsRAN`, `Simu5G`, and `SRS` tools were utilized to simulate network traffic under specific scenarios.

The core experiments of the Doctoral Thesis research were conducted in the Open RAN Testing and Research Laboratory at the Institute of Photonics, Electronics and Telecommunications (IPEC) of Riga Technical University (RTU). An additional experimental study was carried out at the Department of Information and Communication Technologies, Technical University of Cartagena (UPCT) in Spain.

1.6. Scientific Novelty and Main Results

The scientific novelty of the Doctoral Thesis lies in the pioneering work on fine-grained monitoring and classification of network traffic generated by mainstream interactive media applications under heterogeneous network environments. The study achieves high classification performance on multiple metrics with transparent and interpretable processes. The following are the specific novel contributions, supported by 8 publications

1. A robust stacking ensemble learning classifier is proposed that achieved 99 % classification accuracy across 38 categories and an F1-macro score of 0.83 across 101 categories at the flow level on public and proprietary datasets. This model effectively handles datasets with varying sample sizes and severe category imbalance. The Stacking model, introduced in Publication 1, is implemented in the Classification Models section of the Methodology chapter.

2. In the 5G Open RAN environment, it was demonstrated how 5G frequency bands and bandwidth conditions influence video applications traffic distribution. Over 99 % classification accuracy was achieved across 13 categories at the flow level under the enhanced Mobile BroadBand slicing scenario. The corresponding public datasets referenced in Publication 2 are utilised in the Data Acquisition and Preprocessing sections of the Methodology chapter.

3. A novel and explainable *nYFTQC* algorithm has been developed, enabling the construction of 15 proprietary Internet video traffic datasets. These datasets are notable for their completeness, consistency, and transparency. The novel *nYFTQC* algorithm, presented in Publication 3, is applied in the Methodology chapter, in the section on the generation of multi-granularity classification datasets at the flow level.

4. A GCN-based deep learning model is proposed that achieved 86 % classification accuracy across 10 categories at the flow level, incorporating both real-world OTT applications generated network traffic and simulated video traffic using DASH. The GCN-based deep learning model, introduced in Publication 4, is used in the Methodology chapter, in the Classification Models section for performance comparison. Additionally, Publication 4 applies the DASH-based simulated video streaming method in the data acquisition and preprocessing section to generate the streaming VLS simulation datasets.

5. In Publication 5, an improved GNN model was designed for a mixed OTT and XR traffic dataset. Samples were labelled into three QoE-based categories using the Mean Opinion Scores metric, achieving approximately 90 % classification accuracy at the payload level under Wi-Fi network conditions. Publication 5 introduced the mixed traffic dataset generation method, which is applied in the Methodology chapter, in the Data Acquisition and Preprocessing section, for constructing metaverse and cloud gaming datasets. The payload-level classification method, involving byte payload graph sample construction, is also applied in the Methodology chapter, section on the generation of multi-granularity classification datasets for generating payload-level interactive media network traffic datasets.

6. A generative DBN model has been introduced, reaching 92 % classification accuracy across 8 categories at the time series level, where normal OTT applications generated network traffic under the Wi-Fi network environment was mixed with Tor anonymization and VPN

encryption. The same model achieved 94 % accuracy at the flow level on the same dataset. For Publication 6, Tor and VPN-related public datasets were utilized. After data cleaning, these were applied in the Methodology chapter in the Data Acquisition and Preprocessing section to generate the new Tor and VPN sub-dataset within the live streaming dataset.

7. A 1D-CNN model is proposed that achieved 97 % classification accuracy across four categories for the YouTube dataset at the time series level. Based on this model, modified 2D-CNN and RNN models are also developed to expand performance across other granularities. Publication 7 introduced the 1D-CNN model, which was utilized in the Methodology chapter the Classification Models section, for further classification performance comparison.

8. Statistical moment analysis and Fast Fourier Transformation were applied to address packet burstiness and distribution bias at the time series level. Publication 8 introduced methods that improved interpretability but yielded only limited enhancements in classification performance. These methods were still employed to test interactive media network traffic classification performance at the time-series level, as presented in the Generation of Multi-granularity Classification Datasets section of the Methodology chapter.

According to classification results across three granularities for datasets composed of interactive media application network traffic, the main results are as follows:

1. The 1D-CNN model achieved up to 95 % classification accuracy and a macro F1-score of 0.93 among 95 categories at the packet time series level on the Live Streaming dataset with over 40,000 samples.

2. The Stacking model achieved up to 99 % classification accuracy and a macro F1-score of 0.99 among 86 categories at the flow level on the Live Streaming dataset with over 200,000 samples.

3. The 2D-CNN model achieved up to 81 % classification accuracy and a macro F1-score of 0.82 among 95 categories at the payload level on the Live Streaming dataset with over 180,000 samples.

The results achieved during the Doctoral Thesis development have been applied in the following projects:

1. ERDF within the Activity 1.1.1.2 “Postdoctoral Research Aid” of the Specific Aid Objective 1.1.1, No. 1.1.1.2/VIAA/2/18/332, 01.09.2021-30.11.2021.
2. ESF doctoral and academic staff specialization strategic strengthening grant, No. 8.2.2.0/20/I/008, 01.12.2022-30.11.2023.
3. European Union's Recovery and Resilience Facility within the Project No. 5.2.1.1.i.0/2/24/I/CFLA/003, 01.10.2024-30.09.2025.
4. Multipath Roaming Solution for Open RAN (TRANSFER), project No. 1.1.1.3/1/24/A/018, 24.04.2025-30.09.2025.
5. Grant PID2023-148214OB-C21 funded by MICIU/AEI/10.13039/501100011033 and FEDER/EU (Murcia, Spain), 01.05.2023-31.12.2024.

1.7. The Structure of the Doctoral Thesis

The Doctoral Thesis comprises four main chapters. Chapter 1 presents the research context of fine-grained monitoring and classification of interactive media network traffic. It outlines the background, motivation, literature review, objectives, aims, theses, and hypotheses. Chapter 2 describes the methodologies employed to achieve the research goals. It includes five subsections. Chapter 3 provides a detailed analysis of the classification results at the three granularity levels and their corresponding datasets. Chapter 4 concludes the Doctoral Thesis and discusses future research directions.

1.8. Publications and Approbation of the Doctoral Thesis

The results of the Doctoral Thesis are presented in **five** scientific articles and publications in conference proceedings, **four** of which are indexed in SCOPUS, Web of Science (WoS), and Institute of Electrical and Electronics Engineers (IEEE) databases. The author has a total of **eight** publications. The main results of the Doctoral Thesis were summarized in **three** scientific journals. The results of the research were presented at **eight** international scientific conferences.

Scientific articles and publications

1. **Chen, T.**, Grabs, E., Ipatovs, A., Cano, M. "*Ensemble Learning Enabled Flow-level Internet Traffic Classification*" submitted to Journal of Information and Telecommunication (Under review).
2. **Chen, T.**, Benkis, R., Bogdanovs, N., Stetjuha, M., Klūga, J., Jeralovičs, V., Rjazanovs, D., Grabs, E., Ipatovs, A. "*Video Traffic Classification in the 5G Open RAN Network*" submitted to 2025 Photonics & Electromagnetics Research Symposium (PIERS 2025) Conference (Accepted), United Arab Emirates, Abu Dhabi, 4–8 May **2025**.
3. **Chen, T.**, Grabs, E., Ipatovs, A., Cano, M. *A Novel Proprietary Internet Video Traffic Dataset Generation Algorithm*. Applied Sciences, Vol. 15, No. 2, Article number 515. e-ISSN 2076-3417, **2025**.
4. **Chen, T.**, Grabs, E., Ipatovs, A., Pētersons, E., Ancāns, A. *Bitrate-based Video Traffic Classification*. In: 2023 PhotonIcs & Electromagnetics Research Symposium (PIERS 2023): Proceedings, Czech Republic, Prague, 3–6 July **2023**.
5. **Chen, T.**, Grabs, E., Tasic, I., Cano, M. *Network Traffic Analysis for eXtended Reality Applications*. In: XVI Telematics Engineering Conference (JITEL 2023), Spain, Barcelona, 8–10 November **2023**.
6. **Chen, T.**, Grabs, E., Pētersons, E., Efrosinin, D., Ipatovs, A., Bogdanovs, N., Rjazanovs, D. *Multiclass Live Streaming Video Quality Classification Based on Convolutional Neural Networks*. Automatic Control and Computer Sciences, Vol. 54, No. 5, pp. 455–466. ISSN 0146-4116. e-ISSN 1558-108X, **2022**.
7. **Chen, T.**, Grabs, E., Pētersons, E., Efrosinin, D., Ipatovs, A., Klūga, J. *Encapsulated and Anonymized Network Video Traffic Classification with Generative Models*. In: 2022

Workshop on Microwave Theory and Techniques in Wireless Communications 2022 (MTTW'22): Proceedings, Latvia, Riga, 5–7 October **2022**.

8. Grabs, E., **Chen, T.**, Pētersons, E., Efrosinin, D., Ipatovs, A., Klūga, J., Čulkovs, D. *Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks*. In: 2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW 2021), Latvia, Riga, 7–8 October, **2021**.

Presentations at **8** international scientific conferences

1. **Chen, T.**, Benkis, R., Bogdanovs, N., Stetjuha, M., Klūga, J., Jeralovičs, V., Rjazanovs, D., Grabs, E., Ipatovs, A. "*Video Traffic Classification in the 5G Open RAN Network*" submitted to 2025 Photonics & Electromagnetics Research Symposium (PIERS 2025) Conference (Accepted), United Arab Emirates, Abu Dhabi, 4–8 May **2025**.
2. **Chen, T.**, Grabs, E., Tasic, I., Cano, M. *Network Traffic Analysis for eXtended Reality Applications*. In: XVI Telematics Engineering Conference (JITEL 2023), Spain, Barcelona, 8–10 November **2023**.
3. **Chen, T.**, Grabs, E., Ipatovs, A. *Artificial Intelligence Application in Network Traffic Engineering*, 63rd International Scientific Conference of RTU, Latvia, Riga,17 October **2023**.
4. **Chen, T.**, Grabs, E., Ipatovs, A., Pētersons, E., Ancāns, A. *Bitrate-based Video Traffic Classification*. In: 2023 PhotonIcs & Electromagnetics Research Symposium (PIERS 2023): Proceedings, Czech Republic, Prague, 3–6 July **2023**.
5. **Chen, T.**, Grabs, E., Ipatovs, A. *Artificial Intelligence Application in Network Traffic Engineering*, 64th International Scientific Conference of RTU, Latvia, Riga, 14 October **2022**.
6. **Chen, T.**, Grabs, E., Pētersons, E., Efrosinin, D., Ipatovs, A., Klūga, J. *Encapsulated and Anonymized Network Video Traffic Classification with Generative Models*. In: 2022 Workshop on Microwave Theory and Techniques in Wireless Communications 2022 (MTTW'22): Proceedings, Latvia, Riga, 5–7 October **2022**.
7. Grabs, E., **Chen, T.**, Ipatovs, A. *Flow-based Video Quality Traffic Classification for Real-Time SDN Applications*, First Workshop for ERI on Telecommunication and Networks, Romania, Cluj-Napoca, 14–15 March **2022**.
8. Grabs, E., **Chen, T.**, Pētersons, E., Efrosinin, D., Ipatovs, A., Klūga, J., Čulkovs, D. *Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks*. In: 2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW 2021), Latvia, Riga, 7–8 October **2021**.

2. METHODOLOGY

In this chapter, all employed methods are systematically introduced across five main sections.

2.1. Data Acquisition and Preprocessing

Data acquisition and preprocessing are essential steps in constructing standardized training datasets for interactive media applications network traffic analysis. Table 1 summarizes prior knowledge related to network traffic in interactive media applications, consisting of four datasets: Live Streaming, Cloud Gaming, Metaverse, and Streaming VLS Simulation. Each dataset includes multiple sub-datasets. For the Live Streaming dataset, each proprietary sub-dataset corresponds to a specific OTT platform. Traffic data was captured while watching a single video on each platform using *Google Chrome version 100.0.4896.88*. The traffic was recorded as Packet Capture (PCAP) files via *Wireshark 3.4.3* on a *Windows 10 system (Build 18363.1316)*, using an *Intel(R) Wi-Fi 6 AX200 160MHz* adapter in monitor mode. Captured PCAP files were validated using Internet Protocol (IP) and Media Access Control (MAC) addresses, filtered, and truncated to 30 minutes to ensure consistent duration across services. Sub-datasets are categorized by platform name, video resolution, frames per second (FPS), and whether the content was VoD or VLS. Over 84 GB of traffic was collected.

The YouTube sub-dataset exhibited the greatest category diversity. Different OTT platforms recommend various video encoding settings. Most platforms use adaptive bitrate (ABR) streaming to adjust video quality dynamically based on network conditions and device capabilities [109], [20]. During data capture, fixed playback resolutions and latency mode were selected, Wi-Fi connection bandwidth reached over 100 Mbps. However, many platforms, such as YouTube and Vimeo, favor variable bitrate (VBR) encoding, which allows the bitrate to fluctuate for optimal quality [110]–[112]. The usage of VBR encoding is influenced by several factors, including resolution height and width, frame rate, color depth, codec, encoding settings, scene complexity, and motion intensity. As these parameters increase, the required bitrate tends to rise accordingly. In contrast, platforms like Twitch, Facebook, TikTok, and BiliBili often prefer constant bitrate (CBR) encoding [113]–[115]. For VoD applications, downloaded video files can be analyzed to retrieve raw bitrate information. In contrast, for VLS, such data must be inferred indirectly using real-time traffic statistics. This approach is affected by factors such as Content Delivery Network (CDN) cache delays, storage location of video resources, and varying network conditions, including bandwidth, jitter, and packet loss [116].

In the context of 5G networks, traffic characteristics have changed significantly due to different cellular network configuration variances. According to the *5G Coverage Expansion Dataset 1*, *Italian in-lab testbed dataset 1* and *Italian in-lab testbed dataset 2* from the *NANCY* project, video traffic was simulated and captured using `srsRAN` and `PCAPdroid` tools in the *Open RAN* environment. Data was collected across multiple layers of the protocol stack, including MAC, Next Generation Application Protocol (NGAP), Radio Link Control (RLC), and GPRS Tunneling Protocol User Plane (GTP-U), as well as on the edge client side. Different

video files were streamed using various resolutions (480 P, 720 P, 1080 P, 2160 P, and 4320 P), bitrates (6 Mbps, 10 Mbps, 20 Mbps, and 40 Mbps), and bandwidths (10 MHz and 20 MHz) on two frequency bands (*N3* and *N78*) via Real Time Messaging Protocol (RTMP). The resulting constant bitrates list is approximated through rate limiting under various network bandwidth conditions. However, the theoretical maximum transmission rate in 5G networks significantly exceeds the bitrates discussed in this context. Encoding was performed using H.264 and VP8 codecs. These traffic traces are consolidated into the *Big Buck Bunny* sub-dataset under the Live Streaming category [117]–[119].

Besides, even within the same network environment, network traffic distribution varies across different ABR protocols. This variation is influenced by differences in packet buffering policies, latency tolerance, device capabilities, and user behavior. Platforms such as Twitch, Facebook, and TikTok primarily use the HTTP Live Streaming (HLS) protocol. In contrast, BiliBili and Netflix mainly adopt Dynamic Adaptive Streaming over HTTP (DASH), while YouTube and Vimeo support both protocols. HLS was defined by RFC 8216, which enables segmented video transmission. Clients retrieve and play video content through *.m3u8* playlist files and *.ts* segment files [120]. It typically uses the H.264 video codec and supports audio codecs such as AAC, MP3, and AC-3. DASH, standardized by ISO/IEC 23009-1:2022, also supports segmented transmission. Clients use the Media Presentation Description (MPD) playlist file to access *.ts* or *.m4s* segment files. DASH supports a broader range of video codecs, including H.265, H.264, and VP9 [121]. Although many parameters of a video source (e.g., encoding format, resolution, bitrate, framerate, protocol) can be customized before streaming, these details are typically not visible to the client. Each playback instance on an OTT platform exhibits a unique traffic fingerprint based on these hidden parameters. Consequently, traffic characteristics in interactive media applications vary significantly depending on the chosen protocol, platform, and playback scenario.

To eliminate the influence of external environmental parameters on the network traffic distribution of interactive media applications, the characteristics of traffic under different streaming protocols are systematically examined. All servers and clients are configured within the same Local Area Network (LAN) environment to ensure consistent testing conditions. A single Big Buck Bunny video file is used repeatedly in multiple VLS sessions, with different ABR protocols applied. During each session, downstream traffic is captured from the client side, while upstream traffic is recorded from the Simple Realtime Server (`SRS`) using the `Tcpdump` tool [122]. The bidirectional traffic is selected as an individual category. Big Buck Bunny serves as a widely accepted benchmark for video playback testing due to the availability of high-quality raw video formats that support flexible preprocessing for experimental needs [123]. The `SRS` server is deployed in a `Docker` environment and supports several streaming protocols, including RTMP, Web Real-Time Communication (WebRTC), HLS, HTTP FLV live stream (HTTP-FLV), Secure Reliable Transport (SRT), DASH, and GB28181. For this study, five protocols relevant to interactive media applications are selected: RTMP, WebRTC, HLS, HTTP-FLV, and DASH. RTMP is used as the stream pushing protocol, while `FFmpeg` is employed for video editing, transcoding, and segment generation [124]. The duration of each media segment is set to two seconds, and 10 fragments are retained in the sliding window for

each session. All five selected protocols are supported on the client side using playback tools such as `SRS Player`, `VLC`, `Dash JavaScript Player` and `FFplay` [125], [126]. The resulting network traffic from these streaming sessions is compiled into the Streaming VLS Simulation dataset, with a total volume of approximately 20 GB.

The Cloud Gaming and Metaverse datasets exhibit substantial differences in their data characteristics. The Cloud Gaming dataset includes three sub-datasets: *Spitlings* (SP), *Tomb Raider* (TR), and *Thumper* (TH), all of which were captured on *Google's Stadia* platform. Stadia delivers its services through the WebRTC framework, utilizing several protocols including ICE, STUN, TURN, DTLS, RTP, and RTCP [127]. ICE (Interactive Connectivity Establishment) facilitates peer-to-peer communication over UDP in NAT environments by incorporating STUN and TURN. DTLS (Datagram Transport Layer Security) secures datagram-based transmissions. RTP (Real-Time Protocol) is used for media delivery from the server, while RTCP (Real-Time Control Protocol) enables client-side feedback on reception quality. Network traffic was collected under a VBR scenario with no bandwidth limitations. Each sub-dataset is categorized by game title, resolution, codec, game state, and streaming protocol. Due to the unavailability of raw PCAP files, detailed flow-level and payload-level analysis is not feasible [98].

The Metaverse datasets consist of seven distinct services, with network traffic captured using a *virtual desktop streaming* (VDS) configuration. A cloud-based computer served as the rendering platform, hosting the VDS server. An *Oculus Quest 2* device was connected via a VDS client, and traffic was recorded using `Wireshark`. Each session was limited to 15 Mbps bandwidth and had an approximate duration of one minute. Similar to the Cloud Gaming dataset, Metaverse applications also rely on the WebRTC protocol for streaming [108]. In addition to video-related metadata, OTT applications often implement anonymization and encryption techniques. To reflect these characteristics, YouTube and Vimeo traffic samples, which combine Tor anonymization browser and OpenVPN protocol encapsulation encryption, are extracted from the ISCXVPN2016 and ISCXTor2016 public datasets, respectively [29], [30]. The resulting dataset includes four categories; however, it lacks detailed metadata descriptions related to video parameters. To summarize, the network traffic of various interactive media applications has been analyzed based on protocol diversity, video metadata, and network environment characteristics.

Table 1
Summary of Variations in Network Traffic Characteristics of Interactive Media Applications

Datasets	Dataset	Category	Raw PCAP File size	Duration	Network	Source
Live Streaming	YouTube	26	44.8 GB	1800 s	Wi-Fi	Proprietary
	Big_Buck_Bunny	13	10.4 GB	600 s	5G	Public [117], [119]
	BiliBili	15	20.2 GB	1800 s	Wi-Fi	Proprietary
	Twitch	14	8.48 GB	1800 s	Wi-Fi	Proprietary
	Facebook	12	3.78 GB	1800 s	Wi-Fi	Proprietary
	TikTok	6	2.98 GB	1800 s	Wi-Fi	Proprietary
	Vimeo	5	1.84 GB	1800 s	Wi-Fi	Proprietary
	Tor&VPN	4	1.95 GB	945s	Wi-Fi	Public [29], [30]
Cloud Gaming	Spitlings	6	/	600 s	Wi-Fi	Public [98]
	Tomb Raider	5	/	600 s	Wi-Fi	Public [98]
	Thumper	1	/	600 s	Wi-Fi	Public [98]
Metaverse	VRChat	3	338 MB	63 s	Wi-Fi	Public [108]
	TheLab	3	348 MB	63 s	Wi-Fi	Public [108]
	SolarSystemAR	3	345 MB	63 s	Wi-Fi	Public [108]
	RealityMixer	3	350 MB	63 s	Wi-Fi	Public [108]
	Hellblade	3	347 MB	63 s	Wi-Fi	Public [108]
	DiRTRally2.0	3	358 MB	63 s	Wi-Fi	Public [108]
	BigscreenTheatre	3	357 MB	63 s	Wi-Fi	Public [108]
Streaming VLS Simulation	Big_Buck_Bunny	13	4.14 GB	635 s	Ethernet	Proprietary
	Big_Buck_Bunny	13	3.23 GB	635 s	Ethernet	Proprietary
	Big_Buck_Bunny	13	3.17 GB	635 s	Ethernet	Proprietary
	Big_Buck_Bunny	13	3.22 GB	635 s	Ethernet	Proprietary
	Big_Buck_Bunny	13	2.81 GB	635 s	Ethernet	Proprietary
	Big_Buck_Bunny	13	3.23 GB	635 s	Ethernet	Proprietary

Datasets	Dataset	Resolution	Frame rates	Streaming protocol	Bitrate
Live Streaming	YouTube	144p–8K	30; 60	HLS/DASH	VBR/CBR
	Big_Buck_Bunny	480p–8K	/	HLS/DASH	6;10;20;40Mbps
	BiliBili	360p–8K	30; 60; 80; 120	DASH	VBR/CBR
	Twitch	160p–1080p	30; 60	HLS	VBR/CBR
	Facebook	144p–1080p	30	HLS	VBR/CBR
	TikTok	360p–1080p	30; 60	HLS	VBR/CBR
	Vimeo	/	/	HLS/DASH	VBR/CBR
	Tor&VPN	/	/	HLS/DASH	VBR/CBR
Cloud Gaming	Spitlings	720p–4K	/	WebRTC	VBR
	Tomb Raider	720p–4K	/	WebRTC	VBR
	Thumper	1080p	/	WebRTC	VBR
Metaverse	VRChat	/	60; 90; 120	WebRTC	15 Mbps
	TheLab	/	60; 90; 120	WebRTC	15 Mbps
	SolarSystemAR	/	60; 90; 120	WebRTC	15 Mbps
	RealityMixer	/	60; 90; 120	WebRTC	15 Mbps
	Hellblade	/	60; 90; 120	WebRTC	15 Mbps
	DiRTRally2.0	/	60; 90; 120	WebRTC	15 Mbps
	BigscreenTheatre	/	60; 90; 120	WebRTC	15 Mbps
Streaming VLS Simulation	Big_Buck_Bunny	144p–4K	30; 60	HLS	VBR
	Big_Buck_Bunny	144p–4K	30; 60	DASH	VBR
	Big_Buck_Bunny	144p–4K	30; 60	RTMP	VBR
	Big_Buck_Bunny	144p–4K	30; 60	HTTP_FLV	VBR
	Big_Buck_Bunny	144p–4K	30; 60	WebRTC	VBR
	Big_Buck_Bunny	144p–4K	30; 60	RTMP_Upload	VBR

2.2. Interactive Media Network Traffic Patterns

A comprehensive investigation of interactive media traffic patterns is essential for understanding network system performance. This analysis considers traffic data both globally and locally, focusing on statistical, temporal, and queueing-theoretic perspectives. Figure 1 presents packet and payload statistics across various YouTube sub-dataset categories. As video resolution and frame rate increase, the volume of data, measured in both packets and bits, also

rises. This trend is particularly noticeable in 4K and 8K formats, where growth appears exponential. However, due to variability and randomness in video content and playback behavior, traffic characteristics can differ significantly even under identical resolution parameters. These inconsistencies indicate that the increasing relationship is not strictly linear, thus requiring localized and category-specific traffic modeling.

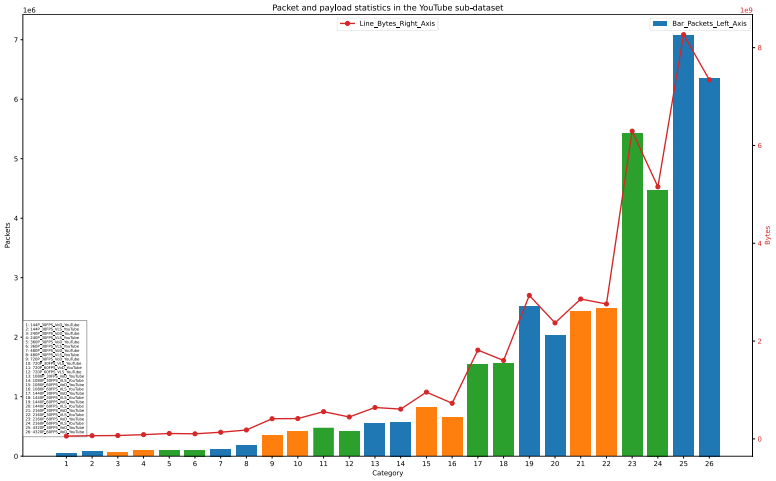


Figure 1. Packet and payload statistics of all categories in the YouTube sub-dataset.

To characterize traffic patterns, especially burstiness, the interarrival time (IAT) between adjacent packets is analyzed using statistical metrics such as mean, minimum, maximum, variance, standard deviation, and coefficient of variation (CV) as shown in Equations (1) – (5). For a vector of packet arrival epoch time $\boldsymbol{t}\ (t_1, t_2, t_3, \dots, t_N)$, then

$$\Delta t_i = t_{i+1} - t_i, i = 1,2, \dots, N-1 \tag{1}$$

$$\boldsymbol{\Delta t} = \mathbf{diff}(\mathrm{t}) = [\mathrm{t}_2 - \mathrm{t}_1,\ \mathrm{t}_3 - \mathrm{t}_2, \dots,\ \mathrm{t}_N - \mathrm{t}_{N-1}] \tag{2}$$

Where $\boldsymbol{\Delta t}$ is a vector of IAT between and $packet_i$ and $packet_{i+1}$, N is total number of packets. When given the interarrival time vector $\boldsymbol{\Delta t} = (\Delta t_1, \Delta t_2, \dots, \Delta t_{N-1})$, next find the mean, standard deviation, and coefficient of variation of the interarrival time vector $\boldsymbol{\Delta t}$.

$$\mu_{\Delta t} = \frac{1}{N-1} \sum_{i=1}^{N-1} \Delta t_i \tag{3}$$

$$\sigma_{\Delta t} = \sqrt{\frac{1}{N-2} \sum_{i=1}^{N-1} (\Delta t_i - \mu_{\Delta t})^2} \tag{4}$$

$$CV_{\Delta t} = \frac{\sigma_{\Delta t}}{\mu_{\Delta t}} \tag{5}$$

The complementary cumulative distribution function (CCDF) is used to visualize the probability of large IAT values. This function illustrates the likelihood that a random variable exceeds a specified threshold and provides insight into the bursty nature of traffic flows. Given a sequence of observed values $\{x_i\}_{i=1}^{N}$, the CCDF is defined as:

$$\mathrm{CCDF}\left(x_{(i)}\right) = 1 - \frac{i}{N}, \quad i = 1,2,\dots,N \tag{6}$$

where $x_{(I)}$ represents the i-th smallest observation, and N is the total number of samples. Assuming each packet has a size b_I (bytes) and is transmitted over a link with rate R = 100 Mbps, the service time for packet i is computed as:

$$s_i = \frac{8 \cdot b_i}{R} \tag{7}$$

The corresponding CCDF denoted as:

$$\mathrm{CCDF}_{\text{interarrival}}\left(\Delta t_{(i)}\right) = 1 - \frac{i}{N-1} \tag{8}$$

$$\mathrm{CCDF}_{\text{service}}\left(s_{(i)}\right) = 1 - \frac{i}{N} \tag{9}$$

Further analysis compares IAT data with heavy-tailed distributions, specifically the Pareto and Weibull distributions. These models capture scenarios where the majority of interarrival times are short, but a few are significantly longer. The Pareto distribution is often used to model systems where a small number of events, such as long delays, have a disproportionate impact. In contrast, the Weibull distribution offers greater flexibility by accommodating both light-tailed and heavy-tailed behavior, depending on its shape parameter [128]. For the vector of filtered interarrival time $\boldsymbol{\Delta t}_{\text{filtered}}$ $\{\Delta t_1, \Delta t_2, \dots, \Delta t_M\}$, the histogram can be computed as:

$$\boldsymbol{\Delta t}_{\text{filtered}} = \{\Delta t_i \in \boldsymbol{\Delta t} \mid \Delta t_i \geq 0.05\} \tag{10}$$

$$h\left(x_j\right) = \{\Delta_{t_i}^{(filtered)} \in \left[x_{j-1}, x_j\right)\} \tag{11}$$

where $h\left(x_j\right)$ represents the count of observations in the interval $\left[x_{j-1}, x_j\right), , j = 1,2,\dots,K$ and K is the number of histogram bins. Divide the counts into each bin by the total number of observations and the width of the corresponding bin:

$$\hat{p}\left(x_j\right) = \frac{h\left(x_j\right)}{M \cdot \Delta x_j} \tag{12}$$

Where $\hat{p}\left(x_j\right)$ is the normalized probability density function (PDF) for the bin x_j, M is the total number of observations, Δx_j is the width of the j-th bin.

Variance plots of aggregated packet counts over different time windows also help identify burstiness. In heavy-tailed traffic scenarios, the variance increases rapidly with larger time

windows. This behavior often follows a power-law pattern, which is a strong indicator of self-similarity or long-range dependence. A steep slope in the variance plot supports the presence of such behavior. The detailed computation process is shown in Algorithm 1.

ALGORITHM 1: VARIANCE OF PACKET COUNTS ACROSS LOG-SCALE WINDOWS

Require: Packet arrival epoch time: $\boldsymbol{t}$ $(t_1, t_2, t_3, \dots, t_N)$ and logarithmically spaced windows: $\mathbf{W} = (-3,1,20)$

Initialize: $V \leftarrow 0$

For $i = 1$ *to* $|\mathbf{W}|$ **do**

$w \leftarrow W[\mathrm{i}]$

Divide $\boldsymbol{t}$ into bins of width w:

$$\text{bins} = \{\mathrm{t}_1, \mathrm{t}_1 + w, \mathrm{t}_1 + 2w, \dots, \mathrm{t}_\mathrm{N}\}$$

Count packets in each bin:

$$\mathrm{c_j} = \#\{\mathrm{t_k} \in [\mathrm{t}_1 + (\mathrm{j}-1)w, \mathrm{t}_1 + \mathrm{j}w)\}$$

Variance: $V[\mathrm{i}] \leftarrow Var(\mathrm{c}_1, \mathrm{c}_2, \dots)$

End For

Ensure Output: Vector $\mathbf{V}$ of variances for each window size

To quantify self-similarity, the Hurst exponent is estimated by analyzing the variance of Daubechies-4 (Db4) wavelet coefficients across multiple scales. The accuracy of this wavelet-based estimation primarily depends on the scaling behavior of the input signal, which reflects its long-range dependence. Key influencing factors include the temporal correlation structure, the statistical properties of the input, such as burstiness or heavy-tailed distributions, and the choice of wavelet, since different wavelets vary in their sensitivity to signal characteristics. A higher variance at larger scales generally indicates persistent behavior and long-range dependence, while a lower variance suggests anti-persistence. A consistent increase in variance across scales is typically associated with self-similar or bursty traffic patterns. In addition, the selection of decomposition levels for linear regression significantly affects the outcome, as very small or large scales may introduce boundary effects or noise. Proper preprocessing, such as removing the mean to ensure stationarity, is also essential to avoid distortions in the estimated scaling behavior. [129].

For system-level performance modeling, the payload size in bytes per packet is used as a proxy for service time, and the packet arrival epoch time is treated as the arrival time. These parameters are incorporated into a GI/G/1 queueing model, which accounts for generalized and potentially bursty or self-similar traffic arrivals as shown in Algorithm 2 [130]. The simulation evaluates key metrics including arrival rate, mean service time, traffic intensity, jitter, queue length, buffer occupancy, and packet loss percentage. The Kingman's approximation is applied to approximate queue behavior in this setting. A sliding window method was used to observe the performance metrics within each window, with the window size set to 10 seconds and a maximum buffer capacity of 700 seconds. Additionally, applying the CCDF to service times reveals the probability of experiencing delays that exceed a specified threshold. This analysis contributes to a deeper understanding of delay distributions and overall system responsiveness.

ALGORITHM 2: TIME SLIDING WINDOW-BASED GI/G/1 QUEUE MODEL ANALYSIS

Require: Arrival time: $\{t_i\}_{i=1}^{N}$, service time: $\{s_i\}_{i=1}^{N}$, window size: w, and buffer limit: B_{limit}

Ensure: Time series vectors of $\lambda_w, \mathbf{S}_w, \rho_w, \mathbf{W}_q, \mathbf{T}_w, \mathbf{J}_w, \mathbf{L}_q, \mathbf{B}_w$, and $\boldsymbol{p}_{loss}$ metrics

Initialize empty vector for each metric

Set $t_{start} \leftarrow \min(\boldsymbol{t})$, $t_{end} \leftarrow \max(\boldsymbol{t})$, prev_delay ← NaN

While $t_{start} < t_{end}$ ***do***

$t_{window} \leftarrow [t_{start}, t_{start} + w)$

Indices of the packets $I = \{i \mid t_i \in t_{window}\}$

If $|I| = 0$ ***then***

$t_{start} \leftarrow t_{start} + w$

Continue

End if

Arrival rate: $\lambda_w \leftarrow \frac{|I|}{w}$

Window service time vector: $\mathbf{s}_w \leftarrow \{s_i \mid i \in I\}$

Mean of service time: $\mathbf{S}_w \leftarrow \text{mean}(\mathbf{s}_w)$

Variance of service time: $E[\mathbf{S}_w{}^2] \leftarrow \text{mean}(\mathbf{s}_w{}^2)$

Coefficient of variation of service time: $C_s^2 \leftarrow \frac{\text{Var}(S)}{(E[S])^2}$

Interarrival time vector: $a_w \leftarrow$ **diff** of packet timestamps in I

Mean interarrival time: $E[A] \leftarrow \text{mean}(a_w)$

Variance of interarrival time: $\text{Var}(A) \leftarrow \text{var}(a_w)$

Coefficient of variation of interarrival time: $C_a^2 \leftarrow \frac{\text{Var}(A)}{(E[A])^2}$

Traffic intensity: $\rho_w \leftarrow \lambda_w \cdot E[S]$

If $\rho_w < 1$ ***then***

Compute waiting time: $W_q \leftarrow E[W_q] \approx \frac{\rho_w}{1-\rho_w} \cdot \frac{C_a^2 + C_s^2}{2} \cdot E[S]$

else

$\mathbf{W}_q \leftarrow \text{NaN}$

End if

Total delay: $\mathbf{T}_w \leftarrow \mathbf{W}_q + \mathbf{S}_w$

If prev_delay is defined, ***then***

$\mathbf{J}_w \leftarrow |\mathbf{T}_w - \text{prev\_delay}|$

else

$\mathbf{J}_w \leftarrow \text{NaN}$

End if

prev_delay ← $\mathbf{T}_w$

Queue length: $\mathbf{L}_q \leftarrow \lambda_w \cdot \mathbf{W}_q$

Buffer size: $\mathbf{B}_w \leftarrow \lambda_w \cdot \mathbf{T}_w$

Packet loss: $\boldsymbol{p}_{loss}{}^{(w)} \leftarrow \frac{L_w}{N_w}$

$t_{start} \leftarrow t_{start} + w$

End While

Return Time series vectors of $\lambda_w, \mathbf{S}_w, \rho_w, \mathbf{W}_q, \mathbf{T}_w, \mathbf{J}_w, \mathbf{L}_q, \mathbf{B}_w$, and $\boldsymbol{p}_{loss}$ metrics

Before analyzing interactive media traffic patterns, appropriate network bandwidths were assigned to different applications. The Live Streaming and Cloud Gaming datasets were configured with 100 Mbps, while the Metaverse dataset used specific bandwidth settings based on their respective scenarios. The Streaming VLS Simulation dataset was collected under an ideal, unrestricted network environment, which makes it unsuitable for system-level performance measurements using the GI/G/1 queueing model. Figure 2 summarizes the statistical, temporal, and system-level performance characteristics of 1080P_60FPS_VLS_YouTube traffic. It is observed that the maximum IAT can reach 4 seconds. The CV for the IAT vector is 18.3464, and the estimated Hurst exponent is 0.8771, all indicating strong burstiness and self-similarity. The histogram of packet interarrivals fits well to both Pareto and Weibull distributions, confirming heavy-tailed behavior. At the system level, the mean traffic intensity is approximately 0.35, and the packet loss rate remains below 5 percent, indicating low system utilization.

Facebook traffic exhibits similar properties, with a high CV of 17.6882 and a clear heavy-tailed distribution reflecting bursty behavior. The packet loss rate exceeds 10 percent, and the instantaneous queue length reaches over 150 packets, as shown in Figure 3. TikTok traffic shows a maximum IAT of less than 0.8 seconds and minimal variance in packet counts. The histogram is concentrated within the first 0.2 seconds of IAT values and does not present a pronounced heavy-tailed distribution. However, the estimated Hurst exponent is still 0.73, which suggests a strong self-similarity, as illustrated in Figure 6.

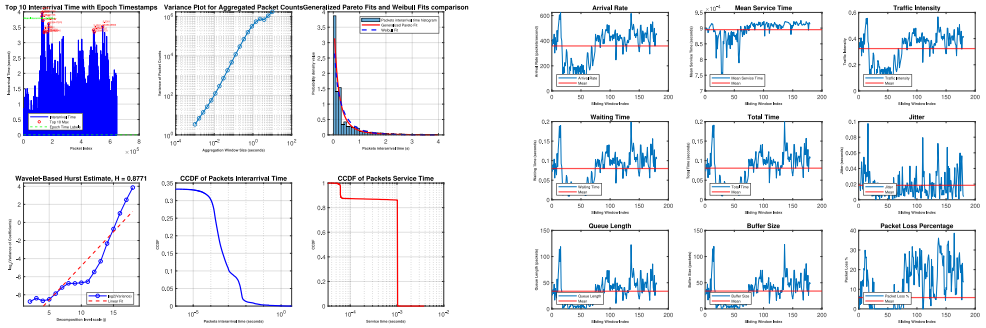


Figure 2. (a) Statistical and temporal characterization of 1080P_60FPS_VLS_YouTube traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

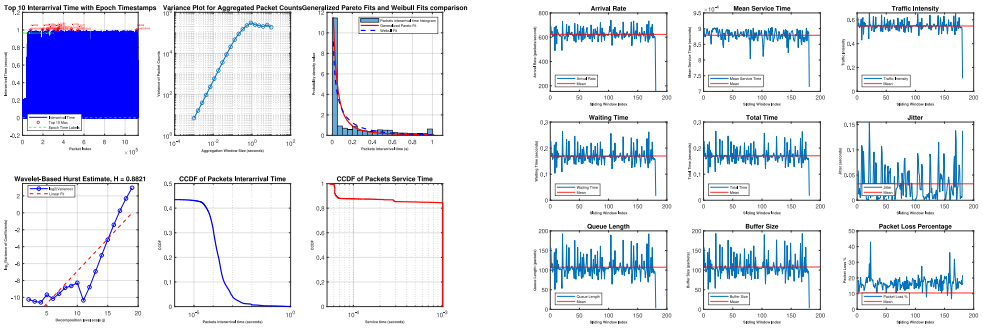


Figure 3. (a) Statistical and temporal characterization of 1080P_30FPS_VLS_Facebook traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

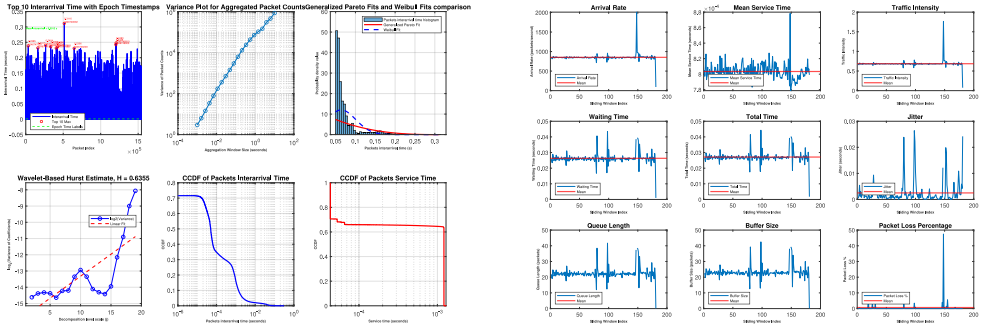


Figure 4. (a) Statistical and temporal characterization of 1080P_60FPS_VLS_Twitch traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

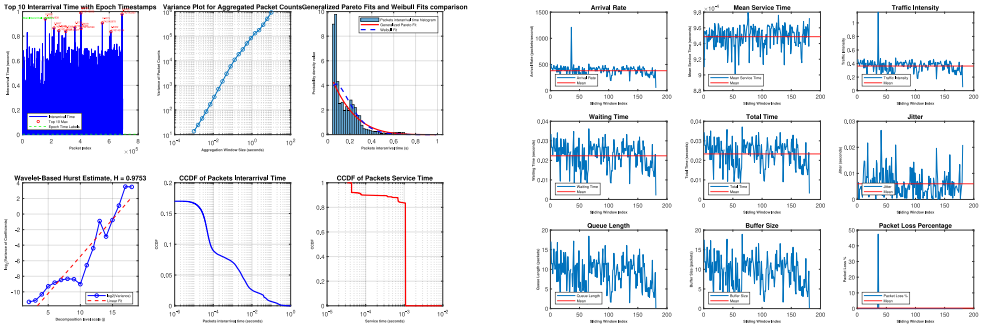


Figure 5. (a) Statistical and temporal characterization of 1080P_60FPS_VLS_BiliBili traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

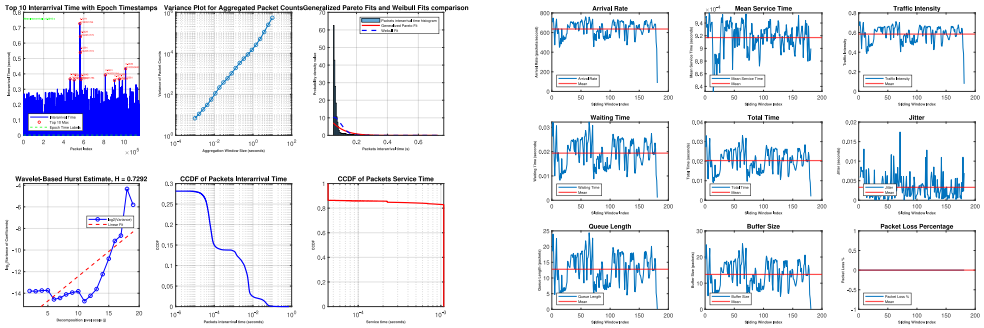


Figure 6. (a) Statistical and temporal characterization of 1080P_60FPS_VLS_TikTok traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

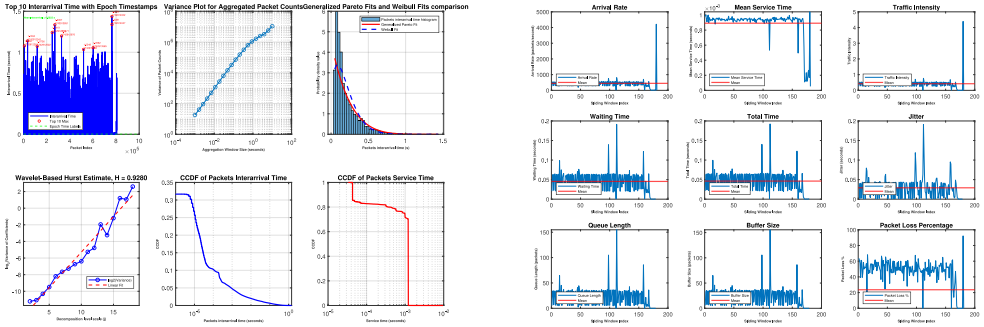


Figure 7. (a) Statistical and temporal characterization of 1080P_30FPS_VoD_Vimeo traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

Twitch traffic has the lowest Hurst exponent at approximately 0.64. Most of its IAT values are under 0.35 seconds. Despite this, the average packet arrival rate is close to 900 packets per second, and the instantaneous traffic intensity exceeds 1, indicating potential pressure on system resources, as shown in Figure 4. Figures 5 and 7 reveal that both BiliBili and Vimeo traffic demonstrate high self-similarity, with Hurst exponents exceeding 0.92. Their interarrival histograms show long heavy-tailed distributions, further validating their bursty nature. In the case of Vimeo, a surge in instantaneous traffic intensity above 1 is observed near the end due to video loop playback, although the average intensity remains around 0.5. In conclusion, the live streaming datasets span multiple OTT applications that use HLS and DASH protocols. These datasets consistently exhibit burstiness and self-similarity, underscoring the need for more advanced modeling and traffic management techniques.

The Cloud Gaming datasets exhibit significantly lower burstiness compared to the Live Streaming dataset. Most adjacent packet IATs are below 0.01 seconds, which is considerably shorter than the average of approximately 0.4 seconds observed in HLS and DASH scenarios. This difference highlights the ultra-low latency advantage of the WebRTC protocol. However,

traffic characteristics vary across different gaming applications. For the SP application, the average packet arrival rate is around 400 packets per second, with a traffic intensity of approximately 0.2, indicating a normal network load. Despite this, the Hurst exponent reaches 0.88, signifying a high degree of self-similarity, as shown in Figure 8. The TH game application shows a similar level of self-similarity, with a Hurst exponent of 0.89. However, the total number of packets transmitted is over seven times greater than that of SP for the same time duration. This results in a mean arrival rate exceeding 2200 packets per second and a traffic intensity greater than 1, as illustrated in Figure 9. For the TR application, the total number of packets is more than 30 percent higher than that of TH, with an average arrival rate exceeding 3000 packets per second, as illustrated in Figure 10. These observations indicate that different cloud gaming applications have varying buffer configurations and source content characteristics. As a result, traffic patterns exhibit high randomness. Appropriate buffering within the network stack helps manage data flow control and congestion, reduces packet loss, and stabilizes transmission rates.

In the case of the Metaverse datasets, all traffic samples are constrained by a maximum bandwidth of 15 Mbps. The average packet arrival rate is close to 800 packets per second, and the mean IAT is less than 0.01 seconds. Therefore, traffic does not display significant burstiness. Nonetheless, the Hurst exponent remains above 0.7, indicating persistent self-similarity. However, because the captured traffic duration is limited to approximately one second, it does not fully reflect the overall traffic patterns of Metaverse applications. In conclusion, the selection of streaming protocols and the configuration of the network environment have a significant impact on traffic patterns. The usage of the WebRTC protocol can effectively reduce burstiness. However, all types of interactive media traffic still exhibit self-similar behavior. Due to the unavailability of pushing stream metadata in real-world interactive media applications, further investigation is necessary to understand how influencing factors affect traffic patterns. The Streaming VLS Simulation datasets are used to focus on statistical and temporal characterization analysis.

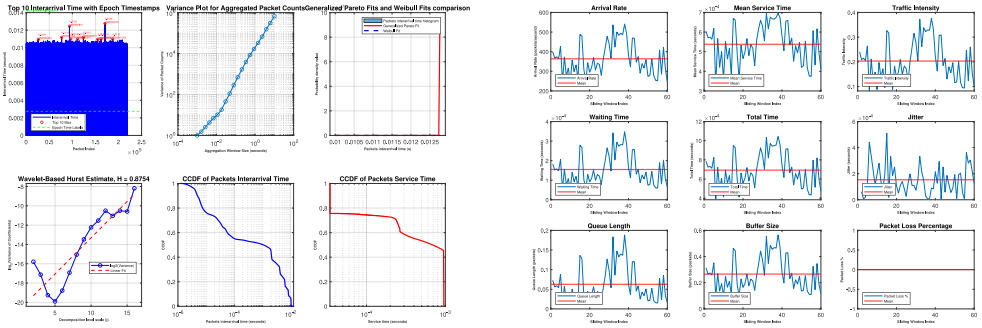


Figure 8. (a) Statistical and temporal characterization of SP_1080P_VP9_RTP traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

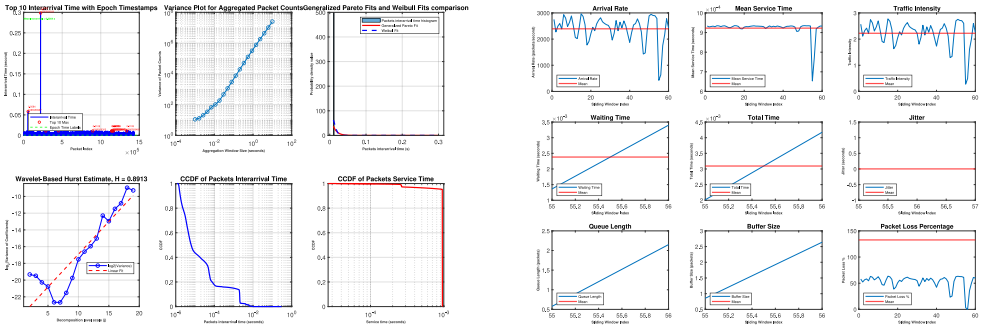


Figure 9. (a) Statistical and temporal characterization of TH_1080P_VP9_RTP traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

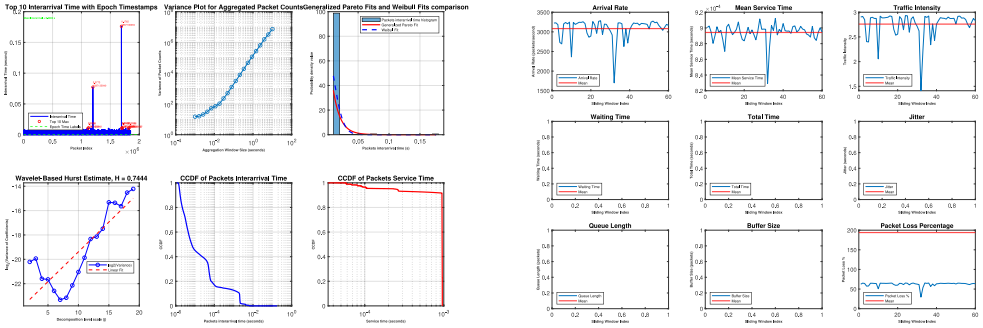


Figure 10. (a) Statistical and temporal characterization of TR_1080P_VP9_RTP traffic. (b) System-level performance metrics based on the GI/G/1 queueing model for the same traffic.

According to the packet and payload statistics summarized in Figure 11, the choice of streaming protocol significantly affects network traffic characteristics. Among the protocols examined, WebRTC exhibits the smallest average payload size and achieves over 20 percent compression efficiency compared to the original video file size, as shown in Table 2, when the resolution is below 2K. However, due to its use of the UDP protocol, WebRTC generates the highest number of packets. UDP transmits data with minimal encapsulation and without congestion control or delivery guarantees, resulting in low latency. This also increases the risk of packet loss and network congestion due to packet flooding. HTTP-FLV also demonstrates high transmission efficiency, with traffic volume reaching approximately 95 percent of the raw video size for resolutions under 2K. It uses persistent HTTP responses to stream continuous FLV data over TCP. Packet payloads often exceed 1500 bytes because of techniques such as TCP Segmentation Offload (TSO) or Generic Segmentation Offload (GSO), which allow the operating system to offload large segments to the network interface card. As a result, capture tools such as Wireshark may display oversized TCP segments.

RTMP provides similar compression efficiency. It multiplexes audio, video, and control data into multiple chunk streams over TCP. While RTMP supports chunking at the application level, many chunk streams in practice carry full message payloads. It often splits large messages into smaller chunks, each transmitted in its own TCP segment. Even with TSO enabled, small

chunks are frequently sent immediately to reduce latency, particularly for audio and control data. This results in many small, frequent TCP packets. Both DASH and HLS are segment-based protocols that support live and on-demand streaming. DASH operates over standard HTTP/1.1 or HTTP/2 using TCP Segments are transmitted via HTTP GET requests, introducing approximately 20 percent overhead compared to the original video size. These segments are often transmitted as small, maximum transmission unit (MTU)-sized packets and reassembled into larger frames in tools like Wireshark. Unlike HTTP-FLV, DASH does not use continuous chunked transfers, and loopback environments may reduce the effectiveness of offloading techniques, resulting in smaller visible packet payloads.

HLS exhibits the highest payload redundancy, exceeding 30 percent for sub-2K resolutions and 50 percent for higher resolutions. This introduces a significant traffic load. HLS typically opens a separate short-lived TCP connection for each media segment, resulting in numerous brief flows with large payloads. In contrast, DASH leverages persistent TCP connections, particularly with HTTP/2, to reduce the number of connections and improve efficiency. While HLS offers broader compatibility with legacy systems, DASH is more flexible and supports advanced features such as multiplexing and low-latency delivery. The repeated use of TCP handshakes, headers, and the lack of connection reuse in HLS contribute to its increased overhead. Persistent connections present a more efficient alternative for modern streaming.

Table 2

Summary of Video Files Parameters in the Streaming VLS Simulation Datasets.

Video	Size	Average Bitrate	I-frames	P-frames	B-frames
144P30FPS	19.8 MB	262 kbps	166	8804	10067
240P30FPS	32.5 MB	430 kbps	165	7411	11461
360P30FPS	53.3 MB	705 kbps	169	6405	12463
480P30FPS	77.9 MB	1029 kbps	169	6252	12616
540P30FPS	89.9 MB	1188 kbps	169	6161	12707
720P30FPS	135 MB	1794 kbps	168	6276	12593
720P60FPS	148 MB	1959 kbps	223	11184	26667
1080P30FPS	263 MB	3481 kbps	149	5952	12928
1080P60FPS	339 MB	4486 kbps	214	10561	27297
1440P30FPS	402 MB	5326 kbps	167	7109	11761
1440P60FPS	427 MB	5650 kbps	226	11219	26629
2160P30FPS	603 MB	7980 kbps	145	5829	13062
2160P60FPS	642 MB	8487 kbps	214	10270	27588

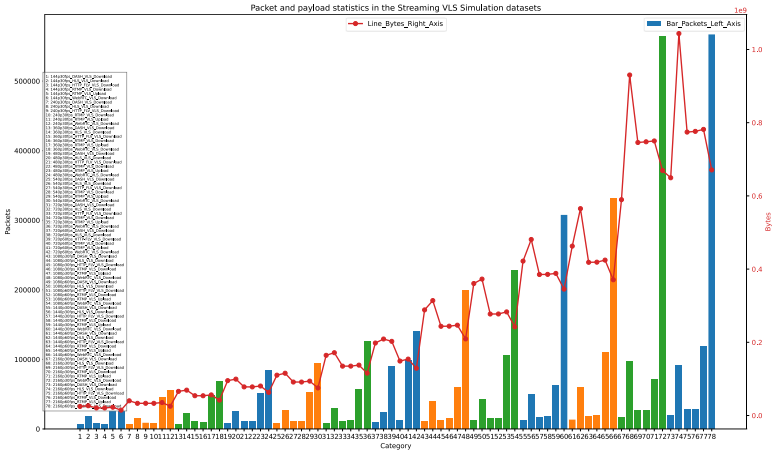


Figure 11. Packet and payload statistics of all categories in the Streaming VLS Simulation datasets.

In addition to streaming protocols, traffic payload characteristics are also influenced by the source video's bitrate and frame structure. As shown in Table 2, the average bitrate increases with resolution and frame rate, ranging from 262 kbps at 144P30FPS to 8487 kbps at 2160P60FPPS. At a fixed resolution, increasing the frame rate from 30 to 60 frames per second approximately doubles the number of frames but results in only a 30% to 60% increase in bitrate. This indicates efficient compression through temporal prediction, likely utilizing inter-frame coding techniques such as prediction from previous frames (P-frames) and prediction from both previous and next frames (B-frames). All analyzed videos are approximately 635 seconds in duration and encoded at multiple resolutions (144P to 2160P) and frame rates (30 FPS and 60 FPS). The encoder maintains a consistent interval between Intra-coded keyframes (I-frames), with keyframes appearing approximately every 3.8 to 4.3 seconds. The corresponding Group of Pictures (GOP) lengths range from 114 to 127 frames for 30 FPS videos and from 170 to 185 frames for 60 FPS videos. These settings suggest an encoding strategy designed to balance random access capabilities with compression efficiency, suitable for both VLS and VoD applications. Higher resolutions and frame rates are associated with a larger proportion of B-frames, particularly in 60 FPS videos where B-frames exceed 70% of all frames. This further enhances compression efficiency and reduces keyframe redundancy. The observed sub-linear scaling of bitrate with respect to resolution and frame rate reflects the use of advanced video codecs such as H.264 or H.265. Given that keyframes contain essential information about bitrate distribution over time, it is important to sample traffic at appropriate time intervals to enable global comparisons of video traffic characteristics.

The raw PCAP file consists of time-ordered packets, each associated with a unique timestamp accurate to six decimal places. These timestamps are discrete and enable precise temporal segmentation. By specifying a fixed sampling interval, the total number of time bins can be determined. Packet counts are then computed for each interval to generate traffic profiles,

as described in Algorithm 3. Figure 12 shows packet distribution over time in the Streaming VLS Simulation dataset, sampled at 100-millisecond intervals using a 1080P resolution and 60 FPS frame rate. When combined with Figure 11, it becomes evident that most packet bursts in the download streams occur between 300 and 600 seconds. In contrast, the upload traffic maintains a relatively stable pattern, with an average of 16.535 packets per 100 milliseconds and minimal fluctuation. The push stream remains stationary at around 20 packets per interval. Burstiness is especially pronounced in traffic using HLS and WebRTC protocols, where the difference between the minimum and maximum packet counts in a 100-millisecond window exceeds 400 packets in some cases. Figure 13 presents the corresponding byte distribution over time under the same conditions, with added keyframe bitrate information. Although payload byte distribution under a specific sampling interval can be obtained using the `Wireshark` *I/O Graphs* module, an alternative method is introduced to provide greater flexibility and precision. This approach relies on packet arrival epoch timestamps to accurately define time bins, allowing summary statistics to be calculated across time-series vectors. These vectors may include packet relative times, payload headers, byte or bit counts, and adjacent packets IAT. The method is applicable to all time-based traffic features and is detailed in Algorithm 4. Figure 13 presents a comparison of byte distribution over time in the Streaming VLS Simulation dataset, sampled at 100-millisecond intervals under 1080P resolution and a 60 FPS frame rate. The figure also incorporates the bitrate distribution of raw video keyframes, enabling a more comprehensive analysis of traffic behavior.

According to Table 2, the 1080P video file used in the simulation spans approximately 633 seconds and contains about 38,070 frames at 60 FPS. The video is encoded using a GOP structure, with an average of 177 frames per GOP and approximately 214 keyframes detected from metadata. `FFmpeg` identified 226 keyframes in total, but bitrate information is available for only 206 of them, indicating some data loss during encoding or extraction. Each GOP typically includes one I-frame followed by around 49 P-frames and 127 B-frames. This structure reflects a high-efficiency compression strategy that maintains visual quality and allows for seeking by periodically refreshing full frames. Keyframes provide full image data and allow direct measurement of bitrate, while P-frames and B-frames use motion prediction and store only differences, making their bitrates less accessible. The streaming process was carried out using `FFmpeg` with the command:

```
ffmpeg -re -i 1080P60FPS.mp4 -c:v libx264 -preset veryfast -r 60 -g 180
```

This command specifies a frame rate of 60 frames per second and sets the GOP size to 180 frames, ensuring consistent intervals between keyframes. This alignment facilitates a clear correlation between encoded video content and observed network traffic behavior. The `-g` parameter enforces a GOP length of approximately 180 frames, around 3 seconds, which closely matches the original video encoding structure. This configuration enhances real-time encoding performance and streaming quality by preserving the regularity of keyframe intervals.

ALGORITHM 3: TRAFFIC INTENSITY CALCULATION OVER SAMPLING INTERVALS

***Input*:** Packet arrival relative time $\mathcal{T} = [t_1, t_2, \dots, t_N]$, sampling interval T_{smp}

Output: Traffic time bins: $\mathcal{T}_{traf} = \left[t_1, t_2, \dots, t_{N_{\text{smp}}}\right]$, Traffic intensity: $\mathcal{I} = \left[i_1, i_2, \dots, i_{N_{\text{smp}}}\right]$

Set number of sampling intervals: $N_{smp} \leftarrow \left\lfloor \frac{\max(\mathcal{T})}{T_{\text{smp}}} \right\rfloor$

For $k = 0$ to $N_{\text{smp}} - 1$

Count packets in each interval $\left[k \cdot T_{\text{smp}}, (k+1) \cdot T_{\text{smp}}\right]$:

$$i_{k+1} \leftarrow \sum_{k=1}^{N} \mathbf{1}_{[k \cdot T_{\text{smp}}, (k+1) \cdot T_{\text{smp}}]}(t_j)$$

Traffic time bins:

$$\mathcal{T}_{traf} \leftarrow \left[T_{\text{smp}}, 2T_{\text{smp}}, \dots, N_{\text{smp}} \cdot T_{\text{smp}}\right]$$

End For

Returning to Figure 13, although a slight offset is observed between the keyframe bitrate distribution and the actual application-layer byte distribution, a consistent burstiness pattern is evident between 300 and 500 seconds. The keyframes during this period also exhibit elevated bitrate characteristics. This observation suggests that, beyond network bandwidth limitations and environmental conditions, the bitrate of raw video keyframes plays a significant role in shaping application-layer traffic patterns. If the streaming source does not implement bitrate control or buffer size constraints before transmission, the resulting traffic may mirror the high-bitrate nature of keyframes, leading to pronounced traffic peaks and bursty behavior. While protocols like WebRTC can mitigate such burstiness due to their low latency and real-time design, they are primarily applicable to interactive streaming scenarios. For VoD applications, WebRTC does not support user-controlled ABR, it relies on network congestion control mechanisms.

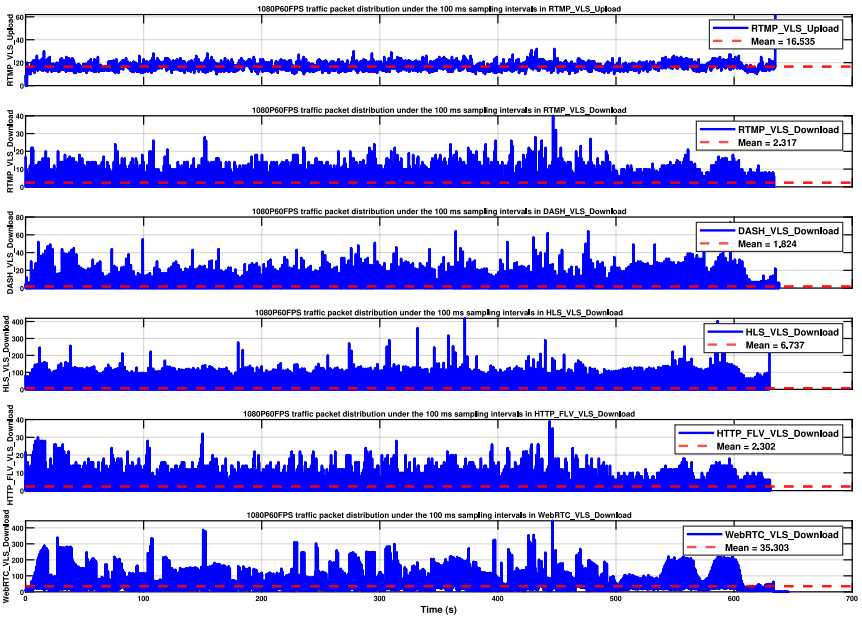


Figure 12. Packet distribution over time in the Streaming VLS Simulation dataset, sampled at 100ms intervals, under 1080P resolution and 60 FPS frame rate.

ALGORITHM 4: TIME SERIES VECTOR OF SPECIFIC SAMPLING INTERVALS GENERATION

***Require*:** Packet arrival epoch time: $\boldsymbol{t}\,(t_1, t_2, t_3, \dots, t_N)$ and sampling interval Δt

***Input*:** Packet arrival relative time $\boldsymbol{\mathcal{T}} = [t_1, t_2, \dots, t_N]$, Packet arrival length $\boldsymbol{\mathcal{P}} = [\mathrm{p}_1, \mathrm{p}_2, \dots, \mathrm{p}_N]$, and Packet arrival delta time $\boldsymbol{\Delta t} = [\delta_1, \delta_2, \dots, \delta_K]$

***Output*:** Byte time series vector under the specific sampling interval, time bins $\mathcal{B}^{(i)}$, and vector sums $\mathcal{S}^{(i)}$ for each $\delta_i \in \boldsymbol{\Delta t}$

For $i \leftarrow 1$ to K

Set current sampling interval: $\delta_i \leftarrow \Delta[i]$

Define bin edges: $\mathcal{E}^{(i)} = \{0, \delta_i, 2\delta_i, \dots, \max(\boldsymbol{\mathcal{T}})\}$

Set number of bins: $M_i \leftarrow \left|\mathcal{E}^{(i)}\right| - 1$

Initialize: $\mathcal{S}^{(i)} \leftarrow 0^{M_i}$

For $j \leftarrow 1\ to\ M_i$

Set bin edges: $\left[e_j^{(i)}, e_{j+1}^{(i)}\right)$

Vector sum: $s_j^{(i)} = \sum_{k=1}^{N} p_k \cdot 1_{\left[e_j^{(i)}, e_{j+1}^{(i)}\right)}(t_k)$

Time bins: $\mathcal{B}^{(i)} = \left[e_1^{(i)}, e_2^{(i)}, \dots, e_{M_i}^{(i)}\right]$

End For

End For

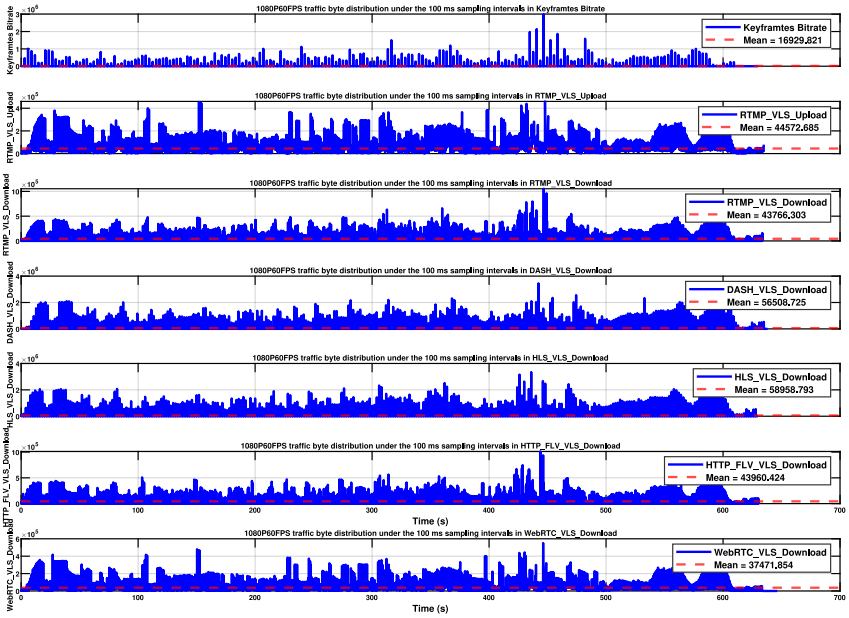


Figure 13. Byte distribution over time in the Streaming VLS Simulation dataset, sampled at 100ms intervals, under 1080P resolution and 60 FPS frame rate.

In summary, interactive media traffic exhibits diverse characteristics. While all observed traffic demonstrates strong self-similarity, burstiness is largely influenced by the source video's bitrate and real-world network conditions. Additionally, the widespread adoption of ABR and advanced video coding techniques has made traffic patterns across different applications more complex and increasingly similar. These observations underscore the demand for fine-grained classification of interactive media traffic to better understand, model, and optimize network behavior in dynamic streaming environments.

2.3. Generation of Multi-Granularity Classification Datasets

In the previous section, Figure 1 presented global packet and payload statistics showing an overall increasing trend corresponding to higher video resolution and frame rates in the YouTube sub-dataset. Although this trend is less apparent for videos below 720P resolution and 60 FPS, combining this analysis with Algorithm 4 reveals that local differences in packet and byte payloads become more pronounced under specific sampling intervals. Figure 14 summarizes the byte payload distribution over 30 minutes in the YouTube sub-dataset, sampled at 100ms intervals. It shows that traffic generated by low resolution and low frame rate applications exhibits notable temporal variability across different time intervals. In the VLS mode, the overall payload volume is slightly higher than in VoD, accompanied by shorter packet IATs. This is attributed to the limited buffer duration in VLS, typically under 20 seconds, compared to VoD, which can buffer over 100 seconds of content under stable network conditions. Consequently, VoD shows longer idle intervals with no packet arrivals.

When the video resolution increases to HD and the frame rate rises to 60 FPS, this behavior becomes more distinct: VLS shows even shorter packet IATs, while VoD exhibits larger IATs, still limited by instantaneous network capability and maximum buffer parameter. Combined with Figure 1, it is evident that higher resolutions and frame rates lead to increased packet and byte payload volumes. Figure 14 further supports this observation by showing a consistent trend in which the mean packet count and byte payload increase with higher video quality. However, when examining the traffic at the packet level, this regularity may be disrupted by the inherent randomness of video transmission. For instance, the 240P_30FPS_VLS traffic data shows a mean packet count of over 86 packets during the 30-minute duration, which is significantly higher than the 26.487 mean packet count observed in the 720P_60FPS_VoD traffic data, as illustrated in Figure 15. This large number of packets does not necessarily correspond to a higher effective payload, as it remains constrained by factors such as the stream pushing mechanism, content encoding format, and the streaming protocol in use. Analysis of the corresponding traffic intensity distributions over the 30-minute interval in Figures 14 and 15 confirms that packet-level and byte-level time series exhibit distinct temporal patterns. These variations serve as valuable features for fine-grained traffic classification tasks, for which time series-based classification approaches have been introduced.

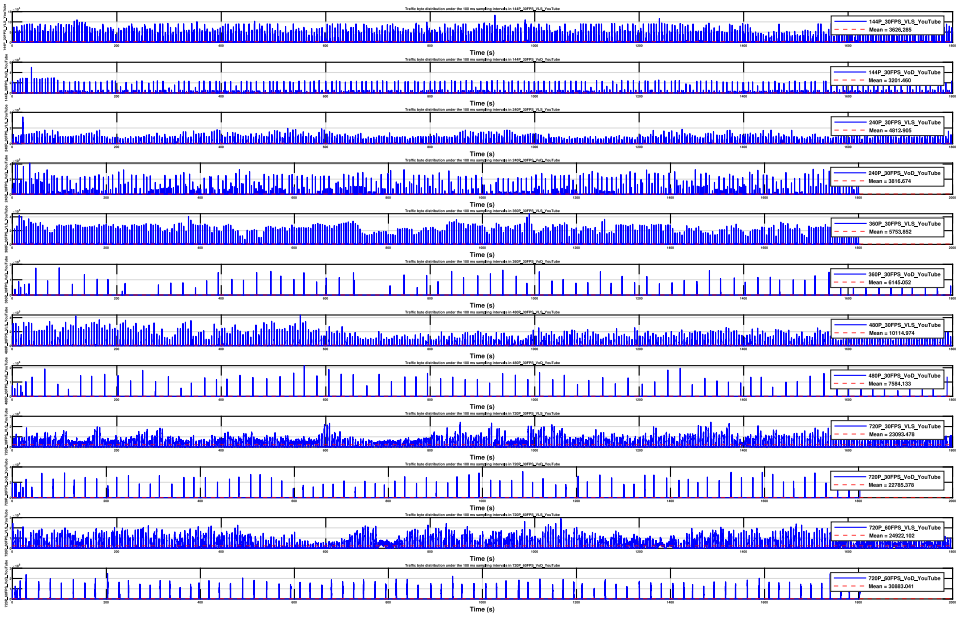


Figure 14. Byte distribution over 30 minutes in the YouTube sub-dataset, sampled at 100ms.

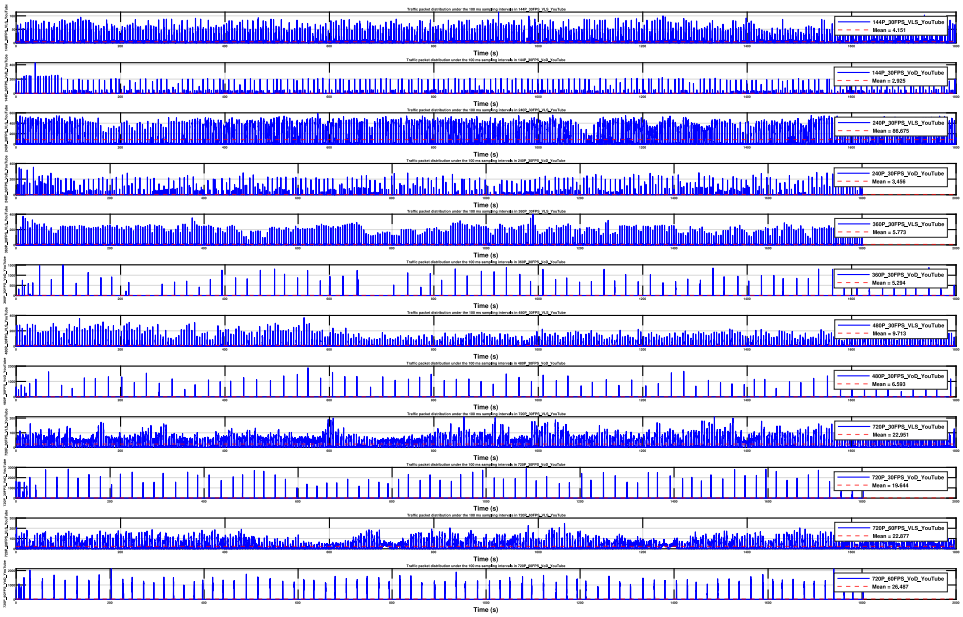


Figure 15. Packet distribution over 30 minutes in the YouTube sub-dataset, sampled at 100ms.

Packet-level and byte-level time series have been widely used to construct training features. These features can be derived from raw traffic byte data, byte matrices, or embedded representations of byte payloads. A single characteristic can develop into multiple temporal and statistical features across different levels and scales [9]. Lin *et al*. proposed a multilevel feature fusion model, MFFusion, which uses vectorized time series consisting of packet and byte information. The model segments each flow into individual packets, retains the time step information, and inputs the sequence into a time series network. This network is built on a stacked bi-directional recurrent neural network (BiRNN), which is designed to capture the temporal characteristics of network flows [131]. In addition to manually engineered features, many deep learning models, such as Autoencoders, Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNN) can all be used to extract deep spatial and temporal features. These models enhance overall classification performance without requiring handcrafted inputs [50], [47], [132]–[135]. The proposed method uses packet-level and byte-level time series with a sliding window approach. Based on a specified time sliding window length, the one-dimensional data is reshaped into feature matrices that summarize either packet or byte-level traffic intensity statistics. Each sample is generated according to the defined sampling interval and window size. Different applications yield distinct feature matrices, which are randomly combined to create complete training datasets with supervised labels. This approach is both simple and efficient, and it has been applied in Publications 7 and 8 [49], [52]. The full implementation procedure is detailed in Algorithm 5.

2.3.1. Time series-level interactive media network traffic datasets

When applying Algorithm 5 to construct time series-level interactive media network traffic datasets, it is essential to select appropriate values for two key hyperparameters: the sampling interval and the sliding window length. Experimental results indicate that although higher sampling intervals can capture more detailed information, they also reduce the total number of feature columns. For example, using a one second sampling interval over a 30-minute traffic duration and a 10-second window results in only about 10 features. Increasing the window length further reduces the number of generated samples by approximately half. To balance the features richness and sample quantity, a sampling interval of 100 milliseconds and a sliding window length of 5 seconds were selected. This configuration yields approximately 50 feature columns. In the Live Streaming datasets, traffic duration varies significantly among sub-datasets, which leads to category imbalance. The Big_Buck_Bunny sub-dataset has a duration of approximately 600 seconds, while the Tor&VPN sub-dataset includes traffic samples of varying lengths. These durations differ considerably from other sub-datasets, which generally span about 30 minutes.

ALGORITHM 5: TIME SERIES-LEVEL DATASETS GENERATION

Require: Packet arrival relative time: $\mathcal{T} = [t_1, t_2, \dots, t_N]$, sampling interval: T_{smp}, sliding window length: T_{wnd}

Ensure: Dataset: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N_R \cdot K} with\ x_i \in R^{N_C},\ y_i \in \{1, \dots, K\}$

Traffic time bins: $\mathcal{T}_{traf} = \left[t_1, t_2, \dots, t_{N_{\text{smp}}}\right]$

Traffic intensity: $\mathcal{I} = \left[i_1, i_2, \dots, i_{N_{\text{smp}}}\right]$

Set number of sampling intervals: $N_{smp} \leftarrow \left\lfloor \frac{\max(\mathcal{T})}{T_{\text{smp}}} \right\rfloor$

Traffic time bins: $\mathcal{T}_{traf} \leftarrow \left[T_{\text{smp}}, 2T_{\text{smp}}, \dots, N_{\text{smp}} \cdot T_{\text{smp}}\right]$

Initialize empty dataset: $D \leftarrow \emptyset$

For **each class** $k = 1$ to K do

Traffic intensity vector $\mathcal{I}^{(k)} = \left[i_1^{(k)}, i_2^{(k)}, \dots, i_{N_{\text{smp}}}^{(k)}\right]$

$N_C \leftarrow \frac{T_{wnd}}{T_{\text{smp}}}$

$N_R \leftarrow \left\lfloor \frac{N_{smp}}{N_C} \right\rfloor$

Reshape: $X^{(k)} \in R^{N_R \times N_C}$ from $\mathcal{I}^{(k)}$

Assign labels: $y^{(k)} \leftarrow [k, \dots, k] \in Z^{N_R}$

End For

Concatenate: $X \leftarrow \text{concat}\left(X^{(1)}, \dots, X^{(K)}\right)$; $y \leftarrow \text{concat}\left(y^{(1)}, \dots, y^{(K)}\right)$.

Update dataset: $\mathcal{D} \leftarrow \text{shuffle}(\{x_i, y_i\})$

After reshaping with a byte-level time series, the Live Streaming dataset contains 31,742 total samples across 95 categories. Most categories have approximately 300 samples. However, the 1080P_30FPS_VoD_YouTube category exceeds 700 samples, while the

4320P_60FPS_VoD_YouTube category contains only about 100, indicating a significant imbalance. A similar issue exists at the packet level, where the 2160P_60FPS_VoD_YouTube category includes around 100 samples, and the 1080P_30FPS_VoD_YouTube category again exceeds 700. In the Cloud Gaming datasets, more than 1400 samples were generated across 12 categories at both packet and byte levels. The Streaming VLS Simulation datasets produced over 10,000 samples across 78 categories, with each category having more than 100 samples. In contrast, the Metaverse dataset is unsuitable for time series-level classification tasks because its raw traffic data lasts only one minute, resulting in approximately 10 samples per category. To mitigate class imbalance, the Synthetic Minority Oversampling TEchnique (SMOTE) was applied along with Edited Nearest Neighbors (ENN) [136]-[137].

SMOTE increases minority class samples by identifying the five nearest neighbors of each sample and generating new synthetic samples through linear interpolation. ENN improves data quality by identifying the three nearest neighbors of majority class samples and removing those that differ from their neighbors. This combined approach balances the dataset by both increasing minority class representation and eliminating noisy majority class samples. After applying SMOTEENN to the Cloud Gaming datasets, the total number of samples increased to over 42,000, representing a 30% gain at both packet and byte levels. However, SMOTEENN may unintentionally reduce the representation of smaller categories and amplify larger ones. After applying SMOTEENN to the Cloud Gaming datasets, the total number of samples remained below 50 percent across the 12 categories at both the packet and byte levels. This occurs because ENN tends to remove samples from sparse categories with conflicting neighbors, while SMOTE generates additional samples for denser and well-represented categories. As a result, rare categories may lose further representation, while majority categories increase in size, especially when their samples are consistent and closely clustered.

2.3.2. Flow-level interactive media network traffic datasets

Flow-level NTC remains one of the most widely adopted and efficient methods, particularly in datacenter routers. Prominent flow management tools such as `NetFlow`, `sFlow`, and `NetStream` exemplify this approach [138]. Contemporary implementations often integrate packet-level and payload-level feature fusion to enhance classification performance [135]. A network flow is typically defined by five key attributes: source IP address, destination IP address, source port, destination port, and protocol type. The foundational work outlined in RFC 2722, published in 1999, formally established the concept of network traffic flows and provided a standardized framework for Internet traffic analysis and management [139]. The core utility of flow-based classification lies in its ability to group packets from different applications according to shared properties, or to represent aggregated event counts over specific periods. A flow may also be considered a sequence of packets. In this context, Xiao *et al.* introduced an Extended Byte Segment Neural Network (EBSNN) neural network architecture that employs an attention encoder layer to extract flow-level features [46]. Similarly, Wu *et al.* proposed the TLS Flow Sequence Network (TFSN), which automatically learns representative features from original TLS flow sequences [47]. Yu *et al.* combined

session-level TLS state transition sequences with packet-level statistical features to effectively capture the diversity of session behaviors and application types [140].

Although many of these methods leverage packet, byte, or time sequence data to construct flow-level samples, they often rely on limited or unitary raw features. In contrast, this study employs the `NFStream` framework to extract 86 flow-level metadata features [141]. These features span multiple scales, including single-directional and bi-directional flow durations, packet and byte counts, and their associated statistical derivative features, offering richer information than traditional unitary descriptors. Additionally, the `NFStream` output is enhanced with the `nDPI` tool, which inspects flow payload headers to identify corresponding protocols and application-level metadata. To integrate all these characteristics, flow-level sample generation is conducted using the *nYFTQC* algorithm, as detailed in Publication 3 [142]. However, in the Big_Buck_Bunny sub-dataset within the Live Streaming datasets, nine categories of raw PCAP files are missing from the UE client side. Consequently, only the gNB user-plane GTP-U layer tunneled PDU packets are available as valid traffic data. Within the GTP-U layer, application data is primarily composed of TCP packets carrying encrypted HTTP payloads. The GTP-U protocol facilitates the transport of user data between the gNB and the core network (UPF). This encapsulation reflects the current industry trend toward HTTPS, driven by demands for privacy, data integrity, and authentication. TCP continues to dominate due to its reliability and congestion control, making it well-suited for securely transmitting encrypted web content. As a result, deep packet inspection within GTP-U tunnels typically reveals a high concentration of TCP traffic with opaque TLS payloads. This pattern underscores the application ecosystem's reliance on secure, connection-oriented communication. However, tools such as `NFStream` are unable to extract metadata from these encapsulated layers or generate the corresponding flow-level samples. Likewise, the `nDPI` tool fails to parse the related protocols and application-level information effectively.

After applying the *nYFTQC* algorithm, it was observed that the Live Streaming dataset produced approximately 108,572 flow samples across 86 categories. However, the dataset remains imbalanced. For instance, the 480P_30FPS_VoD_Twitch category contains the highest number of samples, while the 240P_30FPS_VLS_YouTube category has the fewest, with a difference exceeding 2800 samples. This discrepancy is primarily due to VoD traffic typically using a greater variety of ports and transmitting shorter segments, which leads to the generation of more distinct flows. Additionally, the flow truncation duration is set to one second, which further increases the number of generated flow samples. The total volume of the Live Streaming dataset doubles compared to the original raw dataset. In contrast, the Metaverse dataset generates approximately 4000 flow-level samples across 21 categories, and the Streaming VLS Simulation dataset contains over 50,000 samples spanning 78 categories. Following the application of SMOTEENN, it is similar to time series-level sample resampling, and SMOTEENN reduces the Metaverse dataset to around 600 samples, while the Streaming VLS Simulation dataset expands to approximately three times its original size. Lastly, because the raw PCAP files for the Cloud Gaming dataset are unavailable, flow-level classification could not be conducted for that dataset.

2.3.3. Payload-level interactive media network traffic datasets

Although most interactive media network traffic is already encrypted at the transport layer using SSL/TLS, such as YouTube using the QUIC protocol and Metaverse platforms relying on WebRTC, these protocols are fully encrypted to protect user's privacy. As a result, Deep Packet Inspection (DPI) cannot access or analyze payloads as plain text. However, metadata such as the 20-byte payload header and the TLS Server Name Indication (SNI) extension can still provide useful information for identifying applications or services. Despite these available features, the payload can still be used in classification tasks through embedding methods that extract temporal or spatial features based on deep learning algorithms. Another meaningful approach involves transforming encrypted payloads into graph representations for graph-based classification. For example, Ren *et al*. proposed a Tree-Structured Recurrent Neural Network (Tree-RNN) for traffic classification. In their preprocessing step, they adopted a graph-based method based on the *MNIST IDX* format by splitting raw traffic into individual packets, converting each packet's payload into a grayscale image, reading the bytes, trimming redundancy, and unifying the length to create a structured dataset suitable for graph learning [50]. Similarly, Wang *et al*. used a comparable method to generate grayscale images in 28 by 28 resolution, totaling 784 bytes, by using flow and session-level payload data to fill in the image [44], [143]. Salman *et al*. extracted three features from each packet, including byte length, direction, and timestamp, then normalized these values into three matrices [53]. These were then combined into RGB image channels for classification. All these approaches are based on the concept of payload classification, where the payload may originate from a packet, a flow, a session, or derived features.

The method converts the byte payload of each packet into a grayscale image with a resolution of 100 by 100 pixels, representing 1250 bytes. This configuration captures over 83 percent of the maximum MTU of 1500 bytes. Since most UDP payloads fall below this threshold, shorter payloads are padded with zeros, while longer payloads are truncated to retain only the first 1250 bytes. Each resulting graph sample undergoes z-score normalization to minimize computational overhead. The complete procedure is outlined in Algorithm 6 and it has been verified in Publication 5 [105].

ALGORITHM 6: PAYLOAD-LEVEL DATASETS GENERATION

***Require*:** Traffic PCAP file $\mathcal{P}$, image size (w, h), and maximum samples N

Ensure: Dataset $\mathcal{D} = \{(x_1, y), (x_2, y), \dots, (x_N, y)\}$, where $x_i \in R^{w \times h}$, label $y \in \{1, 2, \dots, K\}$

Initialize empty dataset $D \leftarrow \emptyset$

Extract raw frames: $\mathcal{R} \leftarrow \{r \in \mathcal{P} \mid r \text{ contains payload in each frame}\}$

Extract payloads: $\mathcal{D} \leftarrow \{Raw(r) \mid r \in \mathcal{R}\}$

Sort D in the descending order by payload length $|d|$

Select top N payloads: $\mathcal{D}_{\mathcal{N}} \leftarrow \{d_1, d_2, \dots, d_N\}$

For all $d_i \in \mathcal{D}_{\mathcal{N}}$ ***do***

Convert d_i to byte array $A_i \in R^{n_i}$

Trim to multiple of w: $A_i \leftarrow A_i[0: n_i - (n_i \bmod w)]$

Reshape into matrix: $M_i \in R^{h_i \times w}$

If $h_i < h$ ***then***

Pad M_i with zero rows to reach height h

else

Truncate M_i to first h rows

End if

Normalize: $M_i \leftarrow \left\lfloor 255 \cdot \frac{M_i}{\max(M_i)+\varepsilon} \right\rfloor$, where $\varepsilon \approx 10^{-6}$

Let $x_i \leftarrow M_i$

Update to dataset: $D \leftarrow D \cup \{(x_i, y)\}$

End For

Save each x_i in $\mathcal{D}$ as grayscale PNG image with label y

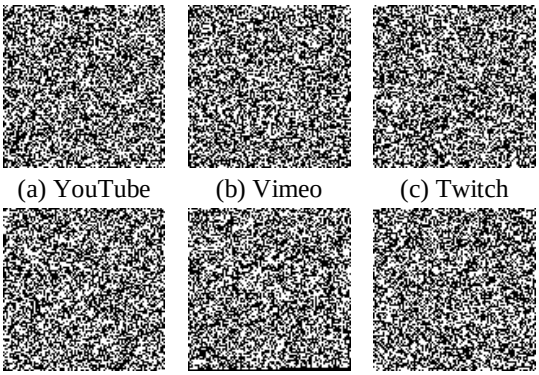
Update dataset: $\mathcal{D} \leftarrow \text{shuffle}(D)$

When applying Algorithm 6 to generate payload-level datasets, significant variations are observed in the sample representations across different datasets. Figure 16 displays a randomly selected graph sample from several interactive media applications. To evaluate their differences, four metrics are used: Euclidean Distance, Cosine Similarity, Mean Squared Error (MSE), and Structural Similarity Index (SSIM). These are implemented using the `scikit-image` and `scikit-learn` Python libraries [144,145]. Euclidean Distance measures the absolute pixel-wise intensity differences. It is well-suited for identifying overall magnitude variations in graph structures or encoded node-edge representations. Cosine Similarity evaluates the angular relationship between two vectorized images, which reflects structural or pattern similarities regardless of magnitude. SSIM assesses visual similarity by examining luminance, contrast, and structural consistency at a local level. This offers a perceptual interpretation of similarity as perceived by a human observer. MSE quantifies the average squared error in pixel intensity, providing a detailed numerical comparison without considering spatial arrangement. Together, these metrics capture a diverse range of insights. Euclidean Distance and MSE reflect numeric intensity differences. Cosine Similarity highlights directional alignment in high-dimensional space. SSIM evaluates how similar graphs appear visually.

Table 3 presents the summary results for each application pair, ordered by Euclidean Distance, Cosine Similarity, MSE, and SSIM. Each value is shown in brackets. Among all comparisons, the Facebook and YouTube pair shows the highest similarity. This is indicated by the lowest Euclidean Distance (30,685.94) and MSE (31,387.57), along with the highest SSIM score (0.04). These results suggest both numerical closeness and visual alignment. Cosine Similarity values remain relatively moderate across most pairs, typically around 0.5. This indicates that most traffic graphs share general directionality in feature space even when pixel-level or visual differences exist. In contrast, BiliBili exhibits the most dissimilar traffic characteristics. It consistently shows high values for Euclidean Distance and MSE, along with negative SSIM scores. For example, the BiliBili-Twitch pair has an SSIM of -0.013, indicating strong structural and perceptual divergence. Most SSIM values across application pairs are close to zero or negative. This confirms that, despite moderate numerical deviations, the visual

and structural layout differs considerably between applications. Facebook and YouTube traffic are the most similar in both numeric and perceptual dimensions. BiliBili differs significantly, possibly due to distinct traffic patterns or encoding strategies. These metrics offer a broad and interpretable comparison of traffic patterns from multiple perspectives. Although the analysis above focuses on differences between entire applications, each application also contains multiple categories with high internal similarity. To address this, the `OpenCV` Python library is used to perform sample feature fusion across different categories [146].

This feature fusion strategy enhances training confidence and supports improved classification performance. Due to the large number of packets in each raw PCAP file, generating byte payload-level representations results in a substantial volume of graph samples. To manage datasets volume and ensure balanced representation, 2000 graph samples are randomly selected from each category to construct the Live Streaming, Metaverse, and Streaming VLS Simulation datasets, respectively. However, the Cloud Gaming datasets lack access to raw PCAP files, making it impossible to generate payload-level network traffic datasets. In summary, considering the practical challenges of original data unavailability, limited sample size, and high similarity among categories, constructing classification datasets at three levels of granularity: time series, flow, and payload, is an effective strategy to support fine-grained classification of interactive media application network traffic.



(a) YouTube (b) Vimeo (c) Twitch

(d) TikTok (d) Facebook (e) BiliBili

Figure 16. Byte payload visualization of different interactive media applications.

Table 3

Summary of Similarity Metrics Differences in Different Interactive Media Applications.

Application	YouTube	Vimeo	Twitch	TikTok	Facebook	BiliBili
YouTube	(0,1,0,1)					
Vimeo	(31090, 0.5, 32219.88, 0.01)	(0,1,0,1)				
Twitch	(31361.89, 0.5, 32785.6, -0.006)	(31083.86,0.5, 32206.88, 0.01)	(0,1,0,1)			
TikTok	(31302.74, 0.5, 32662.06, -0.004)	(31118.36,0.5, 32278.41, 0.01)	(31171.6,0.5, 32388.95,0.01)	(0,1,0,1)		
Facebook	(30685.94, 0.5, 31387.57, 0.04)	(31124.63,0.5, 32291.42, 0.01)	(31008.46,0.5, 32050.82, 0.02)	(31243.48,0.5, 32538.51, 0.005)	(0,1,0,1)	
BiliBili	(31008.46, 0.5, 32050.82, 0.02)	(30973.84,0.5, 31979.3, 0.01)	(31427.13,0.49, 32922.16, -0.013)	(31274.69,0.5, 32603.54, 0.003)	(31455.05,0.49, 32980.68, -0.012)	(0,1,0,1)

2.4. Classification Models

Currently, many state-of-the-art machine learning and deep learning classifiers have been widely adopted for NTC tasks, demonstrating high and robust performance across diverse datasets [13]. Previous works including Publications 1, 2, and Publications 4 to 8 have employed a variety of models, including Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNNs), Deep Belief Networks (DBNs), Extra Trees, Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), and XGBoost [147]–[149], [105], [57], [51], [49], [52]. These models can be broadly categorized into two groups: deep learning models and ensemble learning models. Among deep learning architectures, CNNs are among the most extensively applied for classification. Publications 7 and 8 introduced the 1D-CNN model comprising three one-dimensional convolutional layers followed by four fully connected layers, each with over 1000 neurons. The final layer uses a SoftMax activation function for multi-class classification. A notable aspect of this architecture is the use of progressively smaller kernel sizes (5, 3, and 1) across the convolutional layers, which facilitates the extraction of fine-grained local patterns in sequential data. This design is particularly effective for time series-level and flow-level classification tasks. The corresponding computational process is presented in Equation (13).

$$y_j[t] = \sum_{i=1}^{C_{\text{input}}} \sum_{k=0}^{K-1} x_i[t \times S + k - P] \cdot W_{j,i,k} + b_j \tag{13}$$

Where $y_j[t]$ represents output at time step t for output channel j, $x_i[\cdot]$ represents input from channel i, $W_{j,i,k}$: weight connecting input channel i to output channel j at kernel index k. b_j is bias term for output channel j, C_{input} is the number of input channels, K represents kernel size, S represents stride, and P represents padding.

This architecture was further extended into a 2D-CNN variant by replacing the one-dimensional convolutional layers with two-dimensional counterparts while retaining the overall structure. This adaptation enables the model to exploit spatial feature hierarchies through local receptive fields, offering benefits such as parameter efficiency, translation invariance, and weight sharing. The 2D-CNN is particularly well-suited for payload-level graph classification, where input data is represented in spatial or matrix form. The associated convolutional operation is defined in Equation (14) and shares conceptual similarity with the 1D-CNN.

$$y_j(h,w) = \sum_{i=1}^{C_{input}} \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} x_i\,[h \times S_h + m - P_h, w \times S_w + n - P_w] \cdot W_{j,i,m,n} + b_j \tag{14}$$

Where $y_j(h,w)$ is output value at spatial position (h,w) in output channel j, $x_i[\cdot\cdot]$ represents input feature map from channel i with spatial dimensions (h,w), $W_{j,i,m,n}$: Weight of the kernel at position (m,n) for input channel i and output channel j, K_h, K_w represents the height and width of the convolution kernel, S_h, S_w represents the stride along the height and width dimensions, P_h, P_w represents padding along the height and width dimensions, C_{input} is the number of input channels, and b_j is bias term for output channel j. The 2D-CNN model is modified based on the 1D-CNN model as presented in Table 4.

Table 4
The Architecture Description of the 2D-CNN Model.

Layer	Type	Kernel Size	Filters	Stride	Padding	Input Size	Output Size
Input	-	-	-	-	-	(1, 100, 100)	(1, 100, 100)
Conv1	Conv2D	(5, 5)	32	1	2	(1, 100, 100)	(32, 100, 100)
Pool1	MaxPool2D	(2, 2)	-	2	0	(32, 100, 100)	(32, 50, 50)
Conv2	Conv2D	(3, 3)	64	1	1	(32, 50, 50)	(64, 50, 50)
Pool2	MaxPool2D	(2, 2)	-	2	0	(64, 50, 50)	(64, 25, 25)
Conv3	Conv2D	(1, 1)	128	1	0	(64, 25, 25)	(128, 25, 25)
Flatten	Flatten	-	-	-	-	(128, 25, 25)	(80000)
FC1	Linear	-	1000	-	-	(80000)	(1000)
FC2	Linear	-	500	-	-	(1000)	(500)
FC3	Linear	-	250	-	-	(500)	(250)
FC4	Linear	-	125	-	-	(250)	(125)
Output	Linear	-	C	-	-	(125)	(C)

In addition, an RNN model was developed using three stacked recurrent layers, which were based on the 1D-CNN model. RNNs are effective for modeling temporal dependencies by maintaining internal states over time. Although less complex than architecture such as LSTMs or Transformers, RNNs remain efficient for capturing short-term dependencies in sequential data. The underlying computations are described in Equation (15).

$$h_t = \mathrm{Tanh}(W_{ih} \times x_t + b_{ih} + W_{hh} \times h_{t-1} + b_{hh}) \tag{15}$$

Where h_t in R^H is the hidden state at time step t, $\boldsymbol{x}_t$ in R^D is the input vector at time step t, $\boldsymbol{W}_{ih}$ in $R^{H\times D}$ is the input to the hidden weight matrix, $\boldsymbol{W}_{hh}$ in $R^{H\times H}$ is the hidden to the hidden weight matrix, $\boldsymbol{b}_{ih}$ and $\boldsymbol{b}_{hh}$ in R^H are the respective bias vectors, $\mathrm{Tanh}(\cdot)$ represents element-wise hyperbolic tangent function.

In the flow-level classification task, single flow-level individual feature has relation connections to other statistical or contextual features derived, it could be constructed as graph-structured data, and a GCN model was used. In this approach, a feature dimension from the dataset is used to construct a graph via the k-Nearest Neighbors (k-NN) algorithm, where each node represents a sample and edges reflect neighborhood relationships. This transforms the classification task into a node-level problem. GCNs offer advantages such as the ability to learn from non-Euclidean structures, aggregate information from local neighborhoods, and deliver strong performance in both node and graph classification tasks. Computational formulation is provided in Equation (16).

$$H^{(l+1)} = \mathrm{ReLU}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{16}$$

Where $H^{(l)}$ in $R^{N\times F^{(l)}}$ represents node feature matrix at layer l, $\boldsymbol{W}^{(l)}$ in $R^{F^{(l)}\times F^{(l+1)}}$ represents the trainable weight matrix, $\tilde{A} = A + I$ represents adjacency matrix with self-loops, $\widetilde{\boldsymbol{D}}$ is the diagonal degree matrix of $\tilde{A}$, $ReLU(\cdot)$ represents rectifier activation function.

For classification performance comparison, three benchmark models were selected: Deep Packet, FlowPic, and FS-Net [150]–[152]. Both Deep Packet and FlowPic are designed based on 1D-CNN architectures, while FS-Net employs the Gated Recurrent Unit (GRU), an enhanced recurrent model that incorporates update and reset gates to manage information flow and address the vanishing gradient problem. To further enhance generalization capability, ensemble learning methods were employed. The stacking model includes eleven heterogeneous base learners comprising various boosting and bagging techniques. The final classification output is generated using logistic regression, leveraging the diverse strengths of individual classifiers to improve overall performance as introduced in Publication 1.

Table 4 summarizes the computational complexity of all classification models evaluated. The Stacking model has the highest complexity due to the combined effect of multiple base learners, each involving tree-based computations, along with the overhead introduced by parallel training across all sub-models. The 2D-CNN has relatively high computational and memory costs per batch, particularly when processing high-resolution input and using deep fully connected layers. Both FS-Net and RNN models operate bidirectionally over sequences. Their computational load increases significantly with the number of layers, which makes them relatively demanding. The 1D-CNN model also presents a high computational cost because it uses more than 1000 neurons in the first fully connected layer. In contrast, the computational complexity of the GCN model is primarily determined by the number of graph edges. Since only three neighbors are used to construct each graph, the complexity remains comparatively low. Among all models, Deep Packet and FlowPic exhibit the lowest computational complexity. This is mainly due to their streamlined 1D-CNN architectures and efficient feature extraction mechanisms.

Table 5
Summary of Computational Complexity of All Classification Models.

Models	**Computational Complexity**
1D-CNN [52]	$\mathcal{O}(B \times L \times F_{1D})$
RNN	$\mathcal{O}\left(B \times L \times n_{layers} \times (D \times H + H^2)\right)$
GCN [57]	$\mathcal{O}\left(B \times \left({N \times F_{gcn}}^2 + \lvert E\rvert \times F_{gcn}\right)\right)$
Stacking	$\mathcal{O}(M \times T \times N \times \log N) + \mathcal{O}\big((M + D) \times N\big)$
2D-CNN	$\mathcal{O}(B \times H_s \times W \times F_{2D})$
Deep Packet [150]	$\mathcal{O}(B \times D \times H^2)$
FlowPic [151]	$\mathcal{O}(B \times D \times H^2)$
FS-Net [152]	$\mathcal{O}\big(B \times L \times H^2 \times n_{layers}\big)$

Note:
B − batch size;
C − number of classes;
D − input feature size;
F − number of output channels;
F_{1D} − total filter operations in 1D convolutional layers;
F_{2D} − total filter operations in 2D convolutional layers;
L − length of 1D sequence in 1D-CNN/RNN;
n_{layers} − number of RNN/GRU layers;
F_{gcn} − feature dimension per node;
H − number of hidden units in RNN/GRU layers;
H_s − spatial height of 2D input;
W − spatial width of 2D input;
$|E|$ − number of edges;
N − number of training samples;
M − number of meta learners;
T − number of estimators.

2.5. Experimental Setup

In the previous sections, the statistical characteristics of the multi-granularity interactive media datasets and eight classification models were summarized. Before conducting the classification tasks, several preparatory steps have to be completed. Each dataset is partitioned into three sub-datasets: training, testing, and validation, with the distribution ratio of the total sample size being 60 %, 20 %, and 20 % [144]. Before this partitioning, all dataset samples are randomly shuffled to eliminate any category-based ordering. The training dataset is provided to the classification models alongside the validation dataset. To prevent overfitting, an early stopping mechanism is employed, configured with a patience of 10 and a delta of 0.001. This ensures that training halts automatically once the validation performance ceases to improve beyond the specified threshold. Upon completion of multiple training and validation epochs, the corresponding training and validation accuracies are recorded and denoted as A.Train and A.Validate, respectively.

2.5.1 Feature selection

The next critical step is feature selection. This process often involves advanced feature selection-related algorithms, however, in this case, the strategy is to retain the maximum number of available numerical features to preserve detailed information. For the time series-level datasets, the resulting feature matrix has a shape of 50, and all features are numerical. At the flow-level, a total of 68 numerical features are selected. For the payload-level, each sample consists of a 100 by 100 grayscale image, resulting in 10,000 features after flattening the two-dimensional graph into a one-dimensional vector. Since most classification models are designed to run based on the one-dimensional input, Principal Component Analysis (PCA) is applied to

reduce dimensionality. The number of retained components is set to 100 to minimize computational costs while preserving essential information [153].

2.5.2 Hyperparameters settings

During the training phase, key hyperparameters were selected to optimize model performance and training efficiency. A batch size of 128 was adopted to balance computational throughput with stable gradient estimation. A dropout rate of 20% was applied to reduce overfitting by randomly deactivating neurons during training. All deep learning models were trained for a maximum of 100 epochs. However, early stopping was employed. Training was halted if the validation accuracy failed to improve relative to training accuracy beyond a predefined threshold, thereby conserving resources and preventing overfitting. In each epoch, the models processed mini batches of 128 samples, computing gradients and updating weights through backpropagation. Parameter optimization was performed using Adaptive Moment Estimation (Adam) with an initial learning rate η of 0.0005 as denoted in Equation (23) [154]. Adam combines the advantages of Momentum and RMSProp by adaptively adjusting the effective step sizes for each parameter during training. It does this by maintaining running estimates of the first moment m_t (mean) and second moment v_t (uncentered variance) of the gradients, which are used to compute bias-corrected moment estimates $\hat{m}_t$ and $\hat{v}_t$, where β_1 and β_2 are Adam exponential decay rate constants of 0.9 and 0.999, respectively, and ϵ set as 10^{-8} which is a small constant added to the denominator during the parameter update to avoid division by zero and improve numerical stability. These estimates enable Adam to scale the updates for each parameter individually, allowing for faster convergence and more robust training, especially in the presence of noisy or sparse gradients. Adam's benefits include an internal adaptive mechanism that adjusts the magnitude of parameter updates dynamically throughout the training process, improving optimization efficiency.

The loss function used was *CrossEntropyLoss* in `PyTorch`, which internally combines the log-SoftMax activation and negative log-likelihood (NLL) loss [155]. As denoted in Equation (17), $f(x;\theta)$ represents model function with input x and model parameters θ, it is suitable for multi-class classification tasks. During the training phase, the model generates raw outputs $\boldsymbol{z} = (z_1, z_2, \dots, z_K)$, which are transformed into predicted class probabilities $\hat{y}$ using the SoftMax function, as shown in Equation (24). The loss $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is computed as the negative log-likelihood of the true class label y, penalizing incorrect predictions with high confidence, as shown in Equation (18). To conclude, the integration of *CrossEntropyLoss* and the Adam optimizer forms a robust and effective training pipeline. This combination facilitates faster convergence, adaptivity to different gradient scales, and improved generalization performance across various deep learning architectures.

$$\hat{y} = f(x;\theta) \tag{17}$$

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{18}$$

$$g_t = \nabla_\theta \mathcal{L}(\hat{y}, y) \tag{19}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{20}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{21}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{22}$$

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{23}$$

$$\hat{y}_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)} \quad for \quad k = 1,2,\dots,K \tag{24}$$

Additionally, the Stacking model incorporates eleven heterogeneous meta-learners implemented using three popular ensemble learning libraries: `scikit-learn`, `imbalanced-learn`, `XGBoost`, and `CatBoost` [144], [147], [137], [156], which generate final predictions by training multiple heterogeneous individual learners in parallel. It uses a meta-learning model to combine these predictions through cross-validation. This approach leverages the strengths of various well-performing machine learning models to improve predictive performance. Each of these libraries includes a broad range of hyperparameters. However, due to all meta-learners are based on decision tree-based algorithms, the most critical hyperparameters are the maximum depth of the trees and the number of base estimators. Based on experimental results, a maximum depth of 15 and 500 estimators were selected to prevent overfitting and enhance overall classification accuracy. During the training phase, a five-fold cross-validation strategy was used. This means the entire dataset was randomly split into training, validation, and testing subsets five times. The classification results from each fold were averaged, and the final performance was reported as the mean value across all folds.

2.5.3 Standard score

To reduce computational costs and accelerate training time, z-score standardization is employed as a common normalization method. This technique eliminates the influence of varying dimensional scales across different features, thereby improving the suitability of the data for comparative evaluation and expediting the solution process. Z-score standardization normalizes the variance and converts features into dimensionless quantities, which helps to mitigate the effect of differing units or scales when calculating distances. The computation process is shown in Equation (25), where X_i represents the raw value of a feature for a single sample, μ and σ denote the mean and standard deviation of that feature across all samples, respectively. The resulting standardized value z_i reflects how many standard deviations the original value deviates from the mean. The transformed data conforms to a standard normal distribution, with a mean of 0 and a standard deviation of 1. This normalization method is applied to both the time series-level and flow-level datasets. For the payload-level datasets, however, per-sample z-score normalization is already performed on each graph independently.

$$z_i = \frac{X_i - \mu}{\sigma} \tag{25}$$

2.5.4 Evaluation metrics

For classification performance evaluation, several key metrics are used, including accuracy, precision, recall, and F1-score [157]. Based on the confusion matrix derived from the true and predicted labels, all samples are classified into four categories: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Accuracy is defined as the proportion of correctly classified samples, which include both TP and TN, out of the total number of samples. Recall, also referred to as the true positive rate (TPR), measures the proportion of actual positive samples that are correctly identified. Precision indicates how many of the predicted positive samples are positive. These two metrics are essential for assessing the performance of classification models, particularly in situations involving imbalanced datasets. Equations (27) and (28) provide computations for recall and precision.

The F1-score, which combines precision and recall, is calculated as their harmonic mean. It serves as a critical metric that balances the trade-off between false positives and false negatives. In the case of unbalanced datasets, additional averaging strategies are applied when computing these metrics, including micro, macro, and weighted averages. The micro average treats each individual sample equally across all classes. The macro average calculates the metric for each class independently and then averages the results, assigning equal weight to each class. The weighted average considers the proportion of samples in each class to assign a weight to the metric accordingly. In multi-class classification tasks, weighting is especially important. The symbol w_i denotes the weight assigned to class i, which is calculated as the ratio of the number of samples in class i to the total number of samples across all classes. Equations (29) and (30) provide the specific computation and n refers to the total number of classes in the dataset. To conclude, the accuracy metric is used as the primary testing dataset accuracy metric, denoted as A.Test. Additionally, weighted precision and weighted recall, denoted as P.W and R.W, respectively, are included to address class imbalance issues. To further account for multi-class evaluation, both macro and micro F1-scores are reported, denoted as F.A and F.I, respectively.

$$Accuracy = \frac{\sum_{i=1}^{n} TP_i + TN_i}{\sum_{i=1}^{n} TP_i + TN_i + FP_i + FN_i} \tag{26}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{27}$$

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{28}$$

$$Recall_{weighted} = \frac{\sum_{i=1}^{n} Recall_i \cdot w_i}{|n|} \tag{29}$$

$$Precision_{weighted} = \frac{\sum_{i=1}^{n} Precision_i \cdot w_i}{|n|} \tag{30}$$

$$Recall_{macro} = \frac{\sum_{i=1}^{n} Recall_i}{|n|} \tag{31}$$

$$Precision_{macro} = \frac{\sum_{i=1}^{n} Precision_i}{|n|} \tag{32}$$

$$F1_{macro} = \frac{2 \cdot Precision_{macro} \cdot Recall_{macro}}{Precision_{macro} + Recall_{macro}} \tag{33}$$

$$Recall_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \quad (34)$$

$$Precision_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \quad (35)$$

$$F1_{micro} = \frac{2 \cdot Precision_{micro} \cdot Recall_{micro}}{Precision_{micro} + Recall_{micro}} \quad (36)$$

3. MAIN RESULTS

3.1. Time Series Level Classification Performance

In the time series-level evaluation of network traffic classification performance for interactive media applications, both packet-based and byte-based time series were utilized. Seven classification models were applied, resulting in a total of 28 classification experiments per dataset. These datasets included Live Streaming, Cloud Gaming, and Streaming VLS Simulation. The Metaverse dataset was excluded from time series-level classification due to the short duration of raw traffic data, which prevented the construction of appropriate time series. As summarized in Table 6, the 1D-CNN model achieved the highest classification accuracy on the Live Streaming dataset, reaching 0.65 on both packet and byte time series. Without SMOTEENN resampling, the Stacking model's performance was over 15 % lower than that of the 1D-CNN model. After applying SMOTEENN, classification accuracy for the 1D-CNN model improved by more than 30 %, reaching 0.95 on the packet time series with 95 categories. Similar improvements were observed for all models at the byte time series level.

Notably, the performance gap between the Stacking and 1D-CNN models narrowed significantly after resampling, with the Stacking model achieving classification accuracy within 5 % of that of the 1D-CNN model on both packet and byte time series. Given the large number of categories and imbalanced sample distribution, the macro F1-score was adopted as a more suitable evaluation metric. It treats all categories equally and avoids favoring categories with large sample volumes. The highest macro F1-score, 0.93, was recorded by the 1D-CNN model on the packet time series after SMOTEENN resampling. Table 7 provides detailed per-category classification results. Most categories achieved F1-scores above 0.8. However, some categories still suffered from low precision and recall.

For instance, the 480P_3Mbps_N3_10MHz category, despite its large sample size, had a recall of only 0.68, with many samples misclassified as 1080P_60FPS_VLS_YouTube. These two categories exhibit low traffic similarity, where 480P_3Mbps_N3_10MHz averages around 50 packets per 100 ms, while the 1080P_60FPS_VLS_YouTube category averages around 200 packets per 100 ms. The poor performance is likely due to SMOTE generating interpolated samples in noisy or overlapping regions without considering inter-class distances, leading to misclassifications. Additionally, categories such as 1440P_60FPS_VLS_BiliBili and 1080P_60FPS_VoD_YouTube were also frequently misclassified as 1080P_60FPS_VLS_YouTube due to their high traffic similarity, with average packet counts

around 300 packets per 100 ms. Therefore, low classification performance is not solely caused by small sample sizes, and noise amplification during resampling also plays a significant role.

In the Live Streaming dataset, after applying the SMOTEENN technique, the total number of samples increased by over 25 percent. However, for the originally small-sized and category-balanced Cloud Gaming dataset, the sample count decreased by more than 30 percent, with each category containing fewer than 50 samples. This reduction is attributed to the ENN component of SMOTEENN, which aggressively removes samples, particularly in datasets that are balanced but noisy or exhibit overlapping categories. Although the highest classification accuracy reached 0.94 using the 1D-CNN model at the packet time series level, the limited sample size undermines the reliability of this performance. As shown in Table 9, three categories, namely SP_720P_VP9_WebRTC, TR_2160P_VP9_WebRTC, and TR_1080P_VP9_Play_State, achieved zero correct classifications. This is due to the absence of corresponding category samples in the testing dataset, which significantly reduced both precision and recall. Upon reviewing the raw data files, it is evident that although more than 600 seconds of traffic duration was available to construct the time series level dataset, this duration was still insufficient to support the development of a high-quality and well-balanced dataset.

For the Streaming VLS Simulation dataset, the raw traffic duration was also approximately 600 seconds. However, due to the total sample size exceeding 10,000, the large dataset allowed SMOTE to generate more reliable synthetic samples. Additionally, ENN benefited from the dense neighborhood structure, which reduced the likelihood of false misclassifications. After applying SMOTEENN, all categories were correctly oversampled, and the sample volume for each category more than doubled. According to the classification results presented in Table 10, all models initially achieved classification accuracies below 0.5 at both the packet and byte time series levels without resampling. This highlights the difficulty of accurate classification when the number of samples is insufficient. After applying SMOTEENN, the best performance was achieved by the 1D-CNN model at the byte time series level, which reached a classification accuracy of 0.91 across 78 categories and a macro F1-score of 0.90. Compared to the packet time series level, this result shows over a 7 % improvement in classification performance under the same conditions. This improvement is understandable, as certain categories, such as WebRTC, generate large volumes of packets, contributing to data noise and making classification more challenging at the packet level. The Stacking model also performed well, achieving over 0.8 accuracy, though about 8 % lower than the 1D-CNN. A detailed review of Table 11 shows that the 360P30FPS_HLS_VLS_Download category had a precision of only 0.47, primarily due to frequent misclassification into the 240P30FPS_HLS_VLS_Download category, which had a recall score of 0.59. Similarly, the 480P30FPS_HLS_VLS_Download and 720P60FPS_RTMP_VLS_Upload categories exhibited misclassifications that contributed to lower precision. The 2160P30FPS_HLS_VLS_Download and 2160P60FPS_HLS_VLS_Download categories were also difficult to distinguish due to their high similarity.

Overall, both the 1D-CNN and Stacking models demonstrated strong classification performance at the time series level, with the 1D-CNN model showing the best and most robust

results. Most categories achieved high precision and recall scores, although a few categories had low accuracy due to either model limitations or excessive data noise. However, it is important to note that this high classification performance was largely dependent on the resampling techniques. The effectiveness of these techniques is significantly influenced by several key factors: dataset size, feature variance, class imbalance, and class separability. Larger datasets provide a denser and more representative feature space, allowing SMOTE to generate more meaningful and diverse synthetic samples. High feature variance enables interpolation between distinct samples, reducing the likelihood of ambiguous or noisy points. When genuine class imbalance exists, SMOTE effectively improves minority class representation.

However, in balanced or low-variance datasets, SMOTE may generate low-quality or redundant samples. Additionally, when class separability is poor, synthetic samples may cross decision boundaries, especially when combined with aggressive noise cleaning methods like ENN, which may further degrade data quality by removing valid but overlapping instances. Therefore, the success of synthetic sampling methods is highly dependent on the underlying data characteristics. If the focus shifts to using only raw, real samples without synthetic augmentation, the importance of high-quality data acquisition increases significantly. Active traffic sniffing over extended durations of over 30 minutes may increase collection complexity, but is necessary for sufficient data volume. It is also crucial to consider the quality of packet or byte-level data. For example, traffic from low-resolution and low-framerate sources may consist of a large number of small packets with low payloads, increasing noise and making classification more difficult. Such noise must be carefully cleaned to improve model performance.

Table 6

Classification Performance Summary of Live Streaming Datasets at the Time Series Level

Time series	T_{smp}	T_{wnd}	Resampling	Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
Packet	100 ms	5s	/	1D-CNN	95	0.72	0.65	0.66	0.65	0.65	0.75	0.75
Packet	100 ms	5s	/	RNN	95	0.43	0.42	0.38	0.42	0.41	0.42	0.43
Packet	100 ms	5s	/	GCN	95	0.49	0.43	0.43	0.43	0.43	0.50	0.42
Packet	100 ms	5s	/	Stacking	95	0.60	0.58	0.59	0.58	0.58	0.58	0.58
Packet	100 ms	5s	/	Deep Packet	95	0.57	0.54	0.54	0.54	0.55	0.58	0.57
Packet	100 ms	5s	/	FlowPic	95	0.47	0.46	0.43	0.46	0.46	0.47	0.48
Packet	100 ms	5s	/	FS-Net	95	0.52	0.51	0.50	0.51	0.51	0.54	0.53
Byte	100 ms	5s	/	1D-CNN	95	0.72	0.65	0.66	0.65	0.65	0.77	0.76
Byte	100 ms	5s	/	RNN	95	0.43	0.42	0.39	0.42	0.42	0.42	0.42
Byte	100 ms	5s	/	GCN	95	0.46	0.39	0.41	0.39	0.39	0.52	0.42
Byte	100 ms	5s	/	Stacking	95	0.65	0.63	0.63	0.63	0.63	0.82	0.62
Byte	100 ms	5s	/	Deep Packet	95	0.57	0.55	0.55	0.55	0.55	0.59	0.58
Byte	100 ms	5s	/	FlowPic	95	0.47	0.47	0.44	0.47	0.46	0.48	0.46
Byte	100 ms	5s	/	FS-Net	95	0.55	0.51	0.50	0.51	0.50	0.54	0.54
Packet	100 ms	5s	SMOTEENN	1D-CNN	95	0.96	0.95	0.93	0.95	0.95	0.98	0.97
Packet	100 ms	5s	SMOTEENN	RNN	95	0.75	0.75	0.66	0.75	0.75	0.76	0.76
Packet	100 ms	5s	SMOTEENN	GCN	95	0.84	0.82	0.75	0.82	0.82	0.82	0.79
Packet	100 ms	5s	SMOTEENN	Stacking	95	0.90	0.90	0.82	0.90	0.90	0.99	0.90
Packet	100 ms	5s	SMOTEENN	Deep Packet	95	0.83	0.83	0.75	0.83	0.82	0.84	0.84
Packet	100 ms	5s	SMOTEENN	FlowPic	95	0.73	0.73	0.64	0.73	0.74	0.74	0.74
Packet	100 ms	5s	SMOTEENN	FS-Net	95	0.83	0.82	0.76	0.82	0.82	0.86	0.86
Byte	100 ms	5s	SMOTEENN	1D-CNN	95	0.96	0.94	0.91	0.94	0.94	0.97	0.97
Byte	100 ms	5s	SMOTEENN	RNN	95	0.73	0.73	0.66	0.73	0.73	0.75	0.77

Table 6 (continued)

Byte	100 ms	5s	SMOTEENN	GCN	95	0.83	0.81	0.76	0.81	0.81	0.82	0.78
Byte	100 ms	5s	SMOTEENN	Stacking	95	0.91	0.91	0.84	0.91	0.91	0.99	0.91
Byte	100 ms	5s	SMOTEENN	Deep Packet	95	0.83	0.82	0.76	0.82	0.82	0.84	0.84
Byte	100 ms	5s	SMOTEENN	FlowPic	95	0.72	0.72	0.63	0.72	0.72	0.72	0.73
Byte	100 ms	5s	SMOTEENN	FS-Net	95	0.86	0.85	0.79	0.85	0.85	0.87	0.87

Table 7

Classification Performance in Each Category of Live Streaming Datasets under the 1D-CNN Model after Resampling at the Packet Time Series Level

Category	P.	R.	F.	Category	P.	R.	F.
480P_3Mbps_N3_10Mhz	0.98	0.68	0.8	VPN_Vimeo	1	1	1
480P_3Mbps_N78_10Mhz	0.96	0.91	0.93	VPN_YouTube	0.98	0.78	0.87
480P_3Mbps_N78_20Mhz	0.94	0.92	0.93	160P_30FPS_VoD_Twitch	0.99	0.91	0.95
720P_6Mbps_N3_10Mhz	0.99	0.96	0.97	160P_30FPS_VLS_Twitch	1	0.99	1
720P_6Mbps_N78_10Mhz	0.99	1	1	360P_30FPS_VoD_Twitch	1	0.9	0.95
720P_6Mbps_N78_20Mhz	0.96	0.98	0.97	360P_30FPS_VLS_Twitch	0.97	0.99	0.98
1080P_10Mbps_N3_10Mhz	0.99	1	1	480P_30FPS_VoD_Twitch	1	1	1
1080P_10Mbps_N78_10Mhz	0.98	1	0.99	480P_30FPS_VLS_Twitch	0.97	0.98	0.97
1080P_10Mbps_N78_20Mhz	0.97	0.93	0.95	720P_30FPS_VoD_Twitch	1	1	1
2160P_20Mbps_N3_10Mhz	0.99	0.96	0.98	720P_60FPS_VoD_Twitch	1	1	1
2160P_20Mbps_N78_20Mhz	1	0.99	0.99	720P_30FPS_VLS_Twitch	0.95	0.89	0.92
4320P_40Mbps_N3_10Mhz	0.95	0.97	0.96	720P_60FPS_VLS_Twitch	0.96	0.94	0.95
4320P_40Mbps_N78_20Mhz	0.97	0.99	0.98	1080P_30FPS_VoD_Twitch	0.97	0.96	0.97
2160P_30FPS_VoD_BiliBili	0.99	1	1	1080P_60FPS_VoD_Twitch	0.97	0.98	0.97
2160P_60FPS_VoD_BiliBili	1	0.95	0.98	1080P_30FPS_VLS_Twitch	0.99	1	1
2160P_120FPS_VoD_BiliBili	0.99	1	1	1080P_60FPS_VLS_Twitch	0.95	0.95	0.95
2160P_60FPS_VLS_BiliBili	0.99	0.99	0.99	240P_30FPS_VoD_Vimeo	0.98	0.94	0.96
2160P_80FPS_VLS_BiliBili	0.83	1	0.91	360P_30FPS_VoD_Vimeo	1	0.94	0.97
4320P_30FPS_VoD_BiliBili	1	0.93	0.96	540P_30FPS_VoD_Vimeo	0.77	0.89	0.83
360P_30FPS_VoD_BiliBili	0.93	0.97	0.95	720P_30FPS_VoD_Vimeo	0.95	0.97	0.96
480P_30FPS_VoD_BiliBili	0.98	0.97	0.98	1080P_30FPS_VoD_Vimeo	0.82	0.87	0.84
480P_30FPS_VLS_BiliBili	0.98	0.98	0.98	1440P_30FPS_VoD_YouTube	0.96	0.97	0.96
720P_30FPS_VoD_BiliBili	0.97	1	0.99	1440P_60FPS_VoD_YouTube	0.91	0.82	0.86
720P_30FPS_VLS_BiliBili	0.95	0.92	0.94	1440P_30FPS_VLS_YouTube	0.98	1	0.99
1080P_30FPS_VoD_BiliBili	1	0.9	0.95	1440P_60FPS_VLS_YouTube	0.93	0.97	0.95
1080P_60FPS_VoD_BiliBili	0.96	0.96	0.96	2160P_30FPS_VoD_YouTube	0.9	0.91	0.9
1080P_60FPS_VLS_BiliBili	0.64	0.78	0.7	2160P_60FPS_VoD_YouTube	0.96	1	0.98
1440P_60FPS_VLS_BiliBili	0.75	0.75	0.75	2160P_30FPS_VLS_YouTube	0.99	1	0.99
144P_30FPS_VOD_Facebook	0.87	0.85	0.86	2160P_60FPS_VLS_YouTube	1	0.98	0.99
144P_30FPS_VLS_Facebook	0.8	0.8	0.8	4320P_30FPS_VoD_YouTube	0.99	0.89	0.93
240P_30FPS_VLS_Facebook	0.98	0.96	0.97	4320P_60FPS_VoD_YouTube	0.99	1	1
360P_30FPS_VOD_Facebook	0.79	0.9	0.84	144P_30FPS_VoD_YouTube	0.97	1	0.98
360P_30FPS_VLS_Facebook	0.92	0.63	0.75	144P_30FPS_VLS_YouTube	0.96	0.95	0.95
480P_30FPS_VOD_Facebook	0.97	0.99	0.98	240P_30FPS_VoD_YouTube	0.98	0.97	0.98
480P_30FPS_VLS_Facebook	1	0.8	0.89	240P_30FPS_VLS_YouTube	1	1	1
540P_30FPS_VOD_Facebook	0.92	0.8	0.86	360P_30FPS_VoD_YouTube	1	0.99	1
720P_30FPS_VOD_Facebook	0.95	0.97	0.96	360P_30FPS_VLS_YouTube	0.97	0.97	0.97
720P_30FPS_VLS_Facebook	0.88	0.95	0.91	480P_30FPS_VoD_YouTube	0.98	0.93	0.96
1080P_30FPS_VOD_Facebook	0.98	0.93	0.96	480P_30FPS_VLS_YouTube	0.92	1	0.96
1080P_30FPS_VLS_Facebook	0.87	0.99	0.93	720P_30FPS_VoD_YouTube	0.89	0.73	0.8
360P_30FPS_VLS_TikTok	1	1	1	720P_60FPS_VoD_YouTube	0.95	0.91	0.93
540P_30FPS_VLS_TikTok	1	0.99	1	720P_30FPS_VLS_YouTube	0.94	0.96	0.95
720P_30FPS_VLS_TikTok	0.97	0.98	0.98	720P_60FPS_VLS_YouTube	0.99	0.99	0.99
720P_60FPS_VLS_TikTok	0.99	1	1	1080P_30FPS_VoD_YouTube	0.77	0.83	0.8
1080P_30FPS_VLS_TikTok	1	1	1	1080P_60FPS_VoD_YouTube	1	0.11	0.2
1080P_60FPS_VLS_TikTok	1	0.99	1	1080P_30FPS_VLS_YouTube	0.97	0.99	0.98
Tor_Vimeo	0.95	0.98	0.97	1080P_60FPS_VLS_YouTube	0.38	0.94	0.54
Tor_YouTube	0.97	0.95	0.96				

Table 8

Classification Performance Summary of Cloud Gaming Datasets at the Time Series Level

Time series	T_{smp}	T_{wnd}	Resampling	Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
Packet	100 ms	5 s	/	1D-CNN	12	0.58	0.63	0.55	0.63	0.63	0.61	0.59
Packet	100 ms	5 s	/	RNN	12	0.49	0.53	0.47	0.52	0.50	0.51	0.51
Packet	100 ms	5 s	/	GCN	12	0.30	0.38	0.29	0.38	0.38	0.46	0.37

Table 8 (continued)

Packet	100 ms	5 s	/	Stacking	12	0.65	0.64	0.64	0.64	0.64	0.98	0.61
Packet	100 ms	5 s	/	Deep Packet	12	0.49	0.49	0.45	0.49	0.46	0.49	0.47
Packet	100 ms	5 s	/	FlowPic	12	0.45	0.47	0.41	0.47	0.43	0.43	0.44
Packet	100 ms	5 s	/	FS-Net	12	0.56	0.53	0.51	0.52	0.52	0.51	0.51
Byte	100 ms	5 s	/	1D-CNN	12	0.57	0.65	0.55	0.58	0.52	0.53	0.52
Byte	100 ms	5 s	/	RNN	12	0.56	0.58	0.52	0.53	0.47	0.51	0.48
Byte	100 ms	5 s	/	GCN	12	0.38	0.37	0.31	0.37	0.37	0.48	0.39
Byte	100 ms	5 s	/	Stacking	12	0.50	0.48	0.48	0.48	0.48	0.98	0.48
Byte	100 ms	5 s	/	Deep Packet	12	0.43	0.45	0.41	0.45	0.48	0.48	0.48
Byte	100 ms	5 s	/	FlowPic	12	0.41	0.43	0.38	0.43	0.42	0.45	0.42
Byte	100 ms	5 s	/	FS-Net	12	0.57	0.59	0.51	0.53	0.48	0.49	0.48
Packet	100 ms	5 s	SMOTEENN	1D-CNN	12	0.95	0.97	0.95	0.96	0.94	0.96	0.89
Packet	100 ms	5 s	SMOTEENN	RNN	12	0.76	0.77	0.65	0.74	0.69	0.76	0.65
Packet	100 ms	5 s	SMOTEENN	GCN	12	0.54	0.64	0.33	0.64	0.64	0.76	0.64
Packet	100 ms	5 s	SMOTEENN	Stacking	12	0.74	0.74	0.72	0.73	0.73	0.97	0.73
Packet	100 ms	5 s	SMOTEENN	Deep Packet	12	0.79	0.81	0.71	0.79	0.73	0.76	0.74
Packet	100 ms	5 s	SMOTEENN	FlowPic	12	0.64	0.70	0.59	0.64	0.59	0.61	0.51
Packet	100 ms	5 s	SMOTEENN	FS-Net	12	0.82	0.88	0.80	0.79	0.74	0.75	0.79
Byte	100 ms	5 s	SMOTEENN	1D-CNN	12	0.93	0.90	0.89	0.89	0.88	0.86	0.96
Byte	100 ms	5 s	SMOTEENN	RNN	12	0.87	0.87	0.82	0.85	0.82	0.79	0.79
Byte	100 ms	5 s	SMOTEENN	GCN	12	0.68	0.65	0.42	0.65	0.65	0.81	0.71
Byte	100 ms	5 s	SMOTEENN	Stacking	12	0.63	0.63	0.59	0.63	0.62	0.97	0.61
Byte	100 ms	5 s	SMOTEENN	Deep Packet	12	0.78	0.79	0.67	0.78	0.82	0.78	0.83
Byte	100 ms	5 s	SMOTEENN	FlowPic	12	0.73	0.79	0.67	0.76	0.69	0.66	0.74
Byte	100 ms	5 s	SMOTEENN	FS-Net	12	0.84	0.80	0.72	0.79	0.78	0.77	0.79

Table 9

Classification Performance in Each Category of Cloud Gaming Datasets under the 1D-CNN Model after Resampling at the Packet Time Series Level

Category	P.	R.	F.
SP_2160P_VP9_WebRTC	0.67	0.67	0.67
SP_720P_VP9_WebRTC	0	0	0
SP_1080P_H264_WebRTC	1	1	1
SP_1080P_VP9_WebRTC	0.88	1	0.94
SP_1080P_VP9_RTP	1	1	1
SP_1080P_VP9_Play_State	0.89	1	0.94
TH_1080P_VP9_RTP	1	1	1
TR_2160P_VP9_WebRTC	0	0	0
TR_720P_VP9_WebRTC	1	0.92	0.96
TR_1080P_VP9_WebRTC	1	1	1
TR_1080P_VP9_RTP	1	1	1
TR_1080P_VP9_ Play_State	0	0	0

Table 10

Classification Performance Summary of Streaming VLS Simulation Datasets at the Time Series Level

Time series	T_{smp}	T_{wnd}	Resampling	Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
Packet	100 ms	5 s	/	1D-CNN	78	0.45	0.44	0.39	0.44	0.43	0.56	0.55
Packet	100 ms	5 s	/	RNN	78	0.30	0.33	0.26	0.31	0.30	0.34	0.34
Packet	100 ms	5 s	/	GCN	78	0.34	0.35	0.31	0.35	0.35	0.51	0.35
Packet	100 ms	5 s	/	Stacking	78	0.34	0.34	0.30	0.34	0.34	0.91	0.34
Packet	100 ms	5 s	/	Deep Packet	78	0.34	0.34	0.30	0.34	0.35	0.40	0.38
Packet	100 ms	5 s	/	FlowPic	78	0.29	0.31	0.27	0.31	0.29	0.33	0.34
Packet	100 ms	5 s	/	FS-Net	78	0.33	0.33	0.29	0.33	0.34	0.39	0.39
Byte	100 ms	5 s	/	1D-CNN	78	0.45	0.45	0.41	0.45	0.45	0.69	0.67
Byte	100 ms	5 s	/	RNN	78	0.28	0.28	0.23	0.28	0.27	0.31	0.30

Table 10 (continued)

Byte	100 ms	5 s	/	GCN	78	0.27	0.25	0.22	0.25	0.25	0.46	0.29
Byte	100 ms	5 s	/	Stacking	78	0.33	0.33	0.29	0.33	0.33	0.93	0.32
Byte	100 ms	5 s	/	Deep Packet	78	0.34	0.34	0.31	0.34	0.32	0.40	0.39
Byte	100 ms	5 s	/	FlowPic	78	0.26	0.26	0.22	0.26	0.27	0.30	0.30
Byte	100 ms	5 s	/	FS-Net	78	0.30	0.31	0.25	0.30	0.30	0.34	0.34
Packet	100 ms	5 s	SMOTEENN	1D-CNN	78	0.85	0.84	0.83	0.84	0.84	0.89	0.91
Packet	100 ms	5 s	SMOTEENN	RNN	78	0.41	0.42	0.37	0.41	0.40	0.41	0.40
Packet	100 ms	5 s	SMOTEENN	GCN	78	0.77	0.76	0.74	0.76	0.76	0.78	0.70
Packet	100 ms	5 s	SMOTEENN	Stacking	78	0.83	0.83	0.82	0.83	0.83	0.99	0.84
Packet	100 ms	5 s	SMOTEENN	Deep Packet	78	0.56	0.57	0.54	0.57	0.57	0.60	0.59
Packet	100 ms	5 s	SMOTEENN	FlowPic	78	0.44	0.45	0.41	0.45	0.45	0.45	0.45
Packet	100 ms	5 s	SMOTEENN	FS-Net	78	0.55	0.54	0.51	0.54	0.55	0.56	0.57
Byte	100 ms	5 s	SMOTEENN	1D-CNN	78	0.91	0.91	0.90	0.91	0.91	0.96	0.96
Byte	100 ms	5 s	SMOTEENN	RNN	78	0.42	0.42	0.41	0.42	0.42	0.45	0.43
Byte	100 ms	5 s	SMOTEENN	GCN	78	0.78	0.76	0.76	0.76	0.76	0.80	0.70
Byte	100 ms	5 s	SMOTEENN	Stacking	78	0.82	0.82	0.79	0.82	0.82	0.99	0.82
Byte	100 ms	5 s	SMOTEENN	Deep Packet	78	0.62	0.61	0.59	0.61	0.60	0.65	0.65
Byte	100 ms	5 s	SMOTEENN	FlowPic	78	0.43	0.44	0.42	0.44	0.45	0.47	0.46
Byte	100 ms	5 s	SMOTEENN	FS-Net	78	0.66	0.65	0.64	0.65	0.66	0.70	0.72

Table 11

Classification Performance in Each Category of Streaming VLS Simulation Datasets under the 1D-CNN Model after Resampling at the Byte Time Series Level

Category	P.	R.	F.	Category	P.	R.	F.
144P30FPS_DASH_VLS_Download	0.8	0.74	0.77	720P60FPS_RTMP_VLS_Download	0.93	0.93	0.93
144P30FPS_HLS_VLS_Download	0.97	1	0.98	720P60FPS_RTMP_VLS_Upload	1	0.67	0.8
144P30FPS_HTTP_FLV_VLS_Download	0.93	0.97	0.95	720P60FPS_WebRTC_VLS_Download	0.96	0.88	0.92
144P30FPS_RTMP_VLS_Download	1	0.93	0.96	1080P30FPS_DASH_VLS_Download	0.96	1	0.98
144P30FPS_RTMP_VLS_Upload	0.93	0.93	0.93	1080P30FPS_HLS_VLS_Download	0.96	0.99	0.98
144P30FPS_WebRTC_VLS_Download	0.9	0.99	0.94	1080P30FPS_HTTP_FLV_VLS_Download	0.87	0.93	0.9
240P30FPS_DASH_VLS_Download	0.88	0.95	0.91	1080P30FPS_RTMP_VLS_Download	0.96	0.94	0.95
240P30FPS_HLS_VLS_Download	0.59	0.83	0.69	1080P30FPS_RTMP_VLS_Upload	0.92	0.92	0.92
240P30FPS_HTTP_FLV_VLS_Download	0.88	0.89	0.88	1080P30FPS_WebRTC_VLS_Download	0.91	0.78	0.84
240P30FPS_RTMP_VLS_Download	0.95	0.9	0.93	1080P60FPS_DASH_VLS_Download	0.88	0.96	0.92
240P30FPS_RTMP_VLS_Upload	0.97	0.9	0.94	1080P60FPS_HLS_VLS_Download	1	0.85	0.92
240P30FPS_WebRTC_VLS_Download	0.91	0.91	0.91	1080P60FPS_HTTP_FLV_VLS_Download	0.97	0.9	0.94
360P30FPS_DASH_VLS_Download	0.96	0.62	0.75	1080P60FPS_RTMP_VLS_Download	0.94	0.96	0.95
360P30FPS_HLS_VLS_Download	0.44	0.47	0.46	1080P60FPS_RTMP_VLS_Upload	0.76	0.96	0.85
360P30FPS_HTTP_FLV_VLS_Download	0.85	0.91	0.88	1080P60FPS_WebRTC_VLS_Download	0.87	0.87	0.87
360P30FPS_RTMP_VLS_Download	0.96	0.96	0.96	1440P30FPS_DASH_VLS_Download	0.95	0.88	0.91
360P30FPS_RTMP_VLS_Upload	0.97	0.93	0.95	1440P30FPS_HLS_VLS_Download	0.78	0.82	0.8
360P30FPS_WebRTC_VLS_Download	0.82	0.94	0.88	1440P30FPS_HTTP_FLV_VLS_Download	0.82	0.97	0.89
480P30FPS_DASH_VLS_Download	0.82	0.82	0.82	1440P30FPS_RTMP_VLS_Download	0.85	0.93	0.89
480P30FPS_HLS_VLS_Download	0.79	0.6	0.68	1440P30FPS_RTMP_VLS_Upload	0.97	0.79	0.87
480P30FPS_HTTP_FLV_VLS_Download	0.96	0.73	0.83	1440P30FPS_WebRTC_VLS_Download	0.95	0.88	0.91
480P30FPS_RTMP_VLS_Download	0.97	0.92	0.95	1440P60FPS_DASH_VLS_Download	0.95	0.81	0.88
480P30FPS_RTMP_VLS_Upload	0.99	0.9	0.94	1440P60FPS_HLS_VLS_Download	0.98	1	0.99
480P30FPS_WebRTC_VLS_Download	0.89	0.96	0.92	1440P60FPS_HTTP_FLV_VLS_Download	0.96	0.93	0.95
540P30FPS_DASH_VLS_Download	0.96	0.95	0.95	1440P60FPS_RTMP_VLS_Download	0.96	0.9	0.93
540P30FPS_HLS_VLS_Download	1	1	1	1440P60FPS_RTMP_VLS_Upload	0.94	0.94	0.94
540P30FPS_HTTP_FLV_VLS_Download	0.94	0.99	0.96	1440P60FPS_WebRTC_VLS_Download	0.94	0.91	0.93
540P30FPS_RTMP_VLS_Download	0.98	1	0.99	2160P30FPS_DASH_VLS_Download	0.89	1	0.94
540P30FPS_RTMP_VLS_Upload	0.89	0.92	0.91	2160P30FPS_HLS_VLS_Download	0.67	0.66	0.67
540P30FPS_WebRTC_VLS_Download	0.91	0.98	0.94	2160P30FPS_HTTP_FLV_VLS_Download	0.99	0.84	0.91
720P30FPS_DASH_VLS_Download	0.85	0.98	0.91	2160P30FPS_RTMP_VLS_Download	0.88	0.92	0.9
720P30FPS_HLS_VLS_Download	0.95	0.96	0.96	2160P30FPS_RTMP_VLS_Upload	0.86	0.96	0.91
720P30FPS_HTTP_FLV_VLS_Download	0.78	0.99	0.87	2160P30FPS_WebRTC_VLS_Download	0.91	0.88	0.89
720P30FPS_RTMP_VLS_Download	0.92	0.92	0.92	2160P60FPS_DASH_VLS_Download	0.9	0.93	0.92
720P30FPS_RTMP_VLS_Upload	0.78	0.96	0.86	2160P60FPS_HLS_VLS_Download	0.68	0.78	0.73
720P30FPS_WebRTC_VLS_Download	0.84	0.89	0.87	2160P60FPS_HTTP_FLV_VLS_Download	0.95	0.99	0.97
720P60FPS_DASH_VLS_Download	0.97	0.78	0.87	2160P60FPS_RTMP_VLS_Download	0.98	0.89	0.93
720P60FPS_HLS_VLS_Download	0.99	1	0.99	2160P60FPS_RTMP_VLS_Upload	0.93	0.94	0.93
720P60FPS_HTTP-FLV_VLS_Download	0.96	0.84	0.9	2160P60FPS_WebRTC_VLS_Download	0.94	1	0.97

3.2. Flow-Level Classification Performance

At the flow-level classification task, Table 12 summarizes the classification results for the Live Streaming dataset. The overall classification performance on the raw dataset was poor, with the highest accuracy reaching only 0.4 using the 1D-CNN model. This was largely due to severe class imbalance among categories. Upon investigating individual categories, the 360P_30FPS_VoD_YouTube traffic in the raw PCAP file is primarily transmitted via the QUIC protocol over UDP. It contains four conversation streams corresponding to four dynamic transport layer ports, with each stream lasting over 200 s and one reaching more than 800 s. Similarly, the 480P_30FPS_VoD_YouTube traffic also uses UDP but includes only two conversation streams, meaning the media sender changed ports just twice during the 30-minute session. Using the `NFStream` tool, a new flow sample is triggered either by changes in any of the five-tuple attributes or by a one-minute timer. If a flow lasts longer than one minute, it gets split into a new sample. Due to this, both 360P_30FPS_VoD_YouTube and 480P_30FPS_VoD_YouTube categories generate only 81 flow samples, most of which are created by the time-based truncation. Since the five-tuple attributes rarely change dynamically, this limits the effectiveness of flow-level sample generation. In contrast, the 720P_60FPS_VLS_Twitch category relies on TCP, includes over 12 TCP conversations, and produces more than 2000 flow-level samples. All three categories span approximately 30 min of traffic, but factors such as IP address changes, port switching, or CDN-based load balancing significantly affect the distribution of flow samples.

Moreover, the Tor_Vimeo raw PCAP file has a duration exceeding 3500 s, which is more than double that of the others. Although it includes only one conversation stream, it still generates over 3000 flow-level samples. The primary reason is that Tor aggregates all streams into a single encrypted tunnel directed to the guard relay, rather than establishing separate sockets for each stream. It also reflects high-quality video playback with short inter-arrival times (IATs), meaning that one-second flow containers cannot hold many packets or payloads, thus requiring more containers and resulting in a larger number of flow samples. In summary, the flow sample volume is heavily influenced by the transmission protocol and the duration of the raw PCAP file. After applying SMOTEENN, the 360P_30FPS_VoD_YouTube and 480P_30FPS_VoD_YouTube categories were oversampled more than tenfold, reaching over 1000 samples each, while the original large-volume categories remained unchanged. This significantly reduced class imbalance and improved classification performance. Notably, the Stacking model achieved a macro F1-score exceeding 0.99, outperforming the 1D-CNN by 2 %. According to Table 13, most category samples were correctly classified, with many achieving 100 % accuracy. Even the lowest F1-score was as high as 0.94, demonstrating the robustness and strong performance of the Stacking model.

For the Metaverse dataset, the overall classification performance is observed to be relatively poor, both in the raw dataset and after applying SMOTEENN. According to the results presented in Table 14, the maximum classification accuracy reaches 0.85, and the macro F1-score peaks at 0.73 after resampling. Each category's raw PCAP traffic duration is limited to only one minute. Since Metaverse applications are based on WebRTC and transmitted over

TCP, each PCAP file contains three conversation streams at the transport layer. This results in approximately 180 flow-level samples per category. Due to the small total number of samples, SMOTE struggles to generate meaningful synthetic samples, and ENN tends to aggressively remove data. This is especially problematic in datasets that are balanced but contain noise or exhibit category overlap. As a result, the number of samples per category after SMOTEENN remains under ten, which is insufficient to support effective classification. According to the per-category results in Table 15, more than five categories are entirely missing from the classified results. This is due to the small test set size, which causes some categories to be unrepresented during testing, while misclassifications also remain high. Most categories exhibit F1-scores below 0.8. These findings confirm that small-scale acquisition datasets are not suitable for reliable classification, whether at the time series level or the flow level. Sufficient data volume is essential to enable meaningful classification performance.

According to the classification performance summary of the Streaming VLS Simulation dataset in Table 16, the overall classification results show significant improvement. The Stacking model achieved the highest classification accuracy, reaching 0.9 across 78 categories without applying any resampling techniques. This strong performance is attributed to the uniform duration of approximately 600 s for all raw PCAP traffic files and a relatively balanced category distribution. However, differences in streaming protocols result in varying numbers of flow samples. For instance, using 1080P30FPS video as a reference, the DASH protocol, which is based on TCP and HTTP, generates over 25 TCP conversation streams. The HLS protocol, also built on TCP and HTTP, results in over 642 TCP streams. In contrast, the HTTP-FLV and RTMP protocols, which use persistent session-based TCP connections, each produce only one TCP stream. WebRTC, which is real-time and based on UDP, generates a single UDP stream. DASH and HLS are segmented and pull-based protocols where the client repeatedly downloads small media chunks and playlist files, leading to a high number of TCP connections. DASH initiates a new HTTP request for each 2–10-second video segment, while HLS frequently polls for playlist files and segment files. In comparison, HTTP-FLV and RTMP maintain long-lived TCP sessions without frequent reconnections. WebRTC uses a persistent UDP stream for continuous communication. With all categories having approximately 600 s of traffic, most yield around 600 flow samples. However, DASH and HLS generate more flow samples than other protocols, with increases exceeding 10 percent. Notably, both 2160P30FPS_HLS_VLS_Download and 2160P60FPS_HLS_VLS_Download categories have over 1500 TCP conversation streams, resulting in more than 2000 flow-level samples. In the 2160P60FPS_HLS_VLS_Download case, latency in video decoding extends the playback duration to over 1500 s. This increases inter-arrival times between packets, which causes the flow segmentation process to generate more flow samples even though the actual number of packets remains nearly the same. This kind of time shift introduces noise into the data, which negatively impacts classification performance at the time series level.

After applying SMOTEENN, all category sample counts approximately doubled. With this enhanced dataset, the Stacking model achieved a classification accuracy of 0.98 and a macro F1-score of 0.97. The increased volume and diversity of flow samples also enabled strong performance in other models. For example, the 1D-CNN model achieved a macro F1-score of

0.87, which is about 10 percent lower than the Stacking model. Models such as RNN, Deep Packet, FlowPic, and FS-Net also performed well, although each lagged behind the Stacking model by more than 20 percent. The RNN model had the weakest results, with a classification accuracy of around 0.64 and a macro F1-score of 0.57. As presented in Table 17, the Stacking model maintained F1-scores close to 0.9 for all categories, demonstrating excellent robustness and effectiveness in flow-level classification. In summary, the overall classification performance across the three datasets is excellent after applying resampling and using 86 flow-level training features. However, the small sample volume remains a persistent issue with `NFStream` exported flow data, and alternative flow management tools require further evaluation.

Table 12
Classification Performance Summary of Live Streaming Datasets at the Flow Level

Resampling	Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
/	1D-CNN	86	0.45	0.44	0.39	0.44	0.43	0.56	0.55
/	RNN	86	0.30	0.33	0.26	0.31	0.30	0.34	0.34
/	GCN	86	0.34	0.35	0.31	0.35	0.35	0.51	0.35
/	Stacking	86	0.34	0.34	0.30	0.34	0.34	0.91	0.34
/	Deep Packet	86	0.34	0.34	0.30	0.34	0.35	0.40	0.38
/	FlowPic	86	0.29	0.31	0.27	0.31	0.29	0.33	0.34
/	FS-Net	86	0.33	0.33	0.29	0.33	0.34	0.39	0.39
SMOTEENN	1D-CNN	86	0.98	0.98	0.97	0.98	0.98	0.99	0.99
SMOTEENN	RNN	86	0.95	0.94	0.93	0.94	0.94	0.96	0.96
SMOTEENN	GCN	86	0.89	0.89	0.89	0.89	0.89	0.87	0.87
SMOTEENN	Stacking	86	0.99	0.99	0.99	0.99	0.99	1.00	0.99
SMOTEENN	Deep Packet	86	0.93	0.93	0.92	0.93	0.93	0.94	0.93
SMOTEENN	FlowPic	86	0.87	0.87	0.84	0.87	0.87	0.87	0.87
SMOTEENN	FS-Net	86	0.96	0.96	0.95	0.96	0.96	0.96	0.97

Table 13
Classification Performance in Each Category of Live Streaming Datasets under the Stacking Model after Resampling at the Flow Level

Category	P.	R.	F.	Category	P.	R.	F.
480P_3Mbps_N3_10Mhz	1	1	1	360P_30FPS_VoD_Twitch	0.99	1	1
480P_3Mbps_N78_10Mhz	0.99	0.99	0.99	360P_30FPS_VLS_Twitch	1	1	1
720P_6Mbps_N3_10Mhz	1	0.99	1	480P_30FPS_VoD_Twitch	1	1	1
1080P_10Mbps_N78_10Mhz	0.99	1	1	480P_30FPS_VLS_Twitch	1	1	1
2160P_30FPS_VoD_BiliBili	1	1	1	720P_30FPS_VoD_Twitch	1	1	1
2160P_60FPS_VoD_BiliBili	0.99	1	1	720P_60FPS_VoD_Twitch	1	1	1
2160P_120FPS_VoD_BiliBili	1	1	1	720P_30FPS_VLS_Twitch	1	1	1
2160P_60FPS_VLS_BiliBili	1	1	1	720P_60FPS_VLS_Twitch	1	1	1
2160P_80FPS_VLS_BiliBili	1	1	1	1080P_30FPS_VoD_Twitch	1	1	1
4320P_30FPS_VoD_BiliBili	1	0.99	1	1080P_60FPS_VoD_Twitch	1	1	1
360P_30FPS_VoD_BiliBili	0.99	0.99	0.99	1080P_30FPS_VLS_Twitch	1	1	1
480P_30FPS_VoD_BiliBili	0.99	1	0.99	1080P_60FPS_VLS_Twitch	1	1	1
480P_30FPS_VLS_BiliBili	1	1	1	240P_30FPS_VoD_Vimeo	0.98	0.99	0.98
720P_30FPS_VoD_BiliBili	0.99	0.99	0.99	360P_30FPS_VoD_Vimeo	0.96	0.97	0.96
720P_30FPS_VLS_BiliBili	1	0.99	1	540P_30FPS_VoD_Vimeo	0.97	0.97	0.97
1080P_30FPS_VoD_BiliBili	0.99	0.98	0.99	720P_30FPS_VoD_Vimeo	0.99	0.96	0.97
1080P_60FPS_VoD_BiliBili	1	0.99	1	1080P_30FPS_VoD_Vimeo	1	1	1
1080P_60FPS_VLS_BiliBili	0.99	1	1	1440P_30FPS_VoD_YouTube	0.99	0.99	0.99
1440P_60FPS_VLS_BiliBili	1	1	1	1440P_60FPS_VoD_YouTube	0.92	0.96	0.94
144P_30FPS_VOD_Facebook	1	0.99	0.99	1440P_30FPS_VLS_YouTube	1	0.99	0.99
144P_30FPS_VLS_Facebook	0.99	0.98	0.98	1440P_60FPS_VLS_YouTube	1	1	1
240P_30FPS_VLS_Facebook	0.96	0.99	0.97	2160P_30FPS_VoD_YouTube	1	0.99	1
360P_30FPS_VOD_Facebook	0.98	0.99	0.99	2160P_60FPS_VoD_YouTube	0.97	0.91	0.94
360P_30FPS_VLS_Facebook	0.95	0.95	0.95	2160P_30FPS_VLS_YouTube	0.99	1	0.99

Table 13 (continued)

480P_30FPS_VOD_Facebook	0.99	1	1	2160P_60FPS_VLS_YouTube	1	1	1
480P_30FPS_VLS_Facebook	0.97	0.96	0.97	4320P_30FPS_VoD_YouTube	1	1	1
540P_30FPS_VOD_Facebook	1	1	1	4320P_60FPS_VoD_YouTube	1	1	1
720P_30FPS_VOD_Facebook	1	1	1	144P_30FPS_VoD_YouTube	0.99	1	1
720P_30FPS_VLS_Facebook	1	1	1	144P_30FPS_VLS_YouTube	0.99	1	1
1080P_30FPS_VOD_Facebook	1	1	1	240P_30FPS_VoD_YouTube	0.99	0.99	0.99
1080P_30FPS_VLS_Facebook	1	1	1	240P_30FPS_VLS_YouTube	1	0.99	1
360P_30FPS_VLS_TikTok	1	1	1	360P_30FPS_VoD_YouTube	1	1	1
540P_30FPS_VLS_TikTok	1	1	1	360P_30FPS_VLS_YouTube	1	1	1
720P_30FPS_VLS_TikTok	1	1	1	480P_30FPS_VoD_YouTube	1	1	1
720P_60FPS_VLS_TikTok	1	1	1	480P_30FPS_VLS_YouTube	1	1	1
1080P_30FPS_VLS_TikTok	1	1	1	720P_30FPS_VoD_YouTube	1	1	1
1080P_60FPS_VLS_TikTok	1	1	1	720P_60FPS_VoD_YouTube	1	1	1
Tor_Vimeo	1	1	1	720P_30FPS_VLS_YouTube	0.94	0.96	0.95
Tor_YouTube	1	1	1	720P_60FPS_VLS_YouTube	0.99	0.99	0.99
VPN_Vimeo	1	0.99	0.99	1080P_30FPS_VoD_YouTube	1	1	1
VPN_YouTube	0.99	1	0.99	1080P_60FPS_VoD_YouTube	0.99	0.99	0.99
160P_30FPS_VoD_Twitch	1	1	1	1080P_30FPS_VLS_YouTube	0.97	0.95	0.96
160P_30FPS_VLS_Twitch	1	1	1	1080P_60FPS_VLS_YouTube	0.99	1	1

Table 14

Classification Performance Summary of Metaverse Datasets at the Flow Level

Resampling	Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
/	1D-CNN	21	0.49	0.45	0.45	0.45	0.45	0.56	0.57
/	RNN	21	0.31	0.39	0.34	0.34	0.29	0.39	0.33
/	GCN	21	0.38	0.35	0.35	0.35	0.35	0.43	0.36
/	Stacking	21	0.47	0.46	0.45	0.46	0.46	1.00	0.48
/	Deep Packet	21	0.37	0.36	0.37	0.36	0.37	0.40	0.40
/	FlowPic	21	0.32	0.30	0.31	0.30	0.30	0.35	0.32
/	FS-Net	21	0.40	0.37	0.36	0.37	0.36	0.43	0.40
SMOTEENN	1D-CNN	21	0.87	0.87	0.73	0.85	0.84	0.87	0.89
SMOTEENN	RNN	21	0.63	0.67	0.40	0.63	0.60	0.68	0.71
SMOTEENN	GCN	21	0.80	0.77	0.66	0.77	0.77	0.94	0.80
SMOTEENN	Stacking	21	0.54	0.53	0.51	0.53	0.53	1.00	0.51
SMOTEENN	Deep Packet	21	0.64	0.66	0.39	0.60	0.61	0.62	0.58
SMOTEENN	FlowPic	21	0.49	0.66	0.33	0.58	0.46	0.51	0.47
SMOTEENN	FS-Net	21	0.79	0.80	0.53	0.76	0.72	0.80	0.79

Table 15

Classification Performance in Each Category of Metaverse Datasets under the 1D-CNN Model after Resampling at the Flow Level

Category	P.	R.	F.
BigscreenTheatre_60FPS	0.75	0.6	0.67
BigscreenTheatre_90FPS	0.75	0.9	0.82
BigscreenTheatre_120FPS	0.5	1	0.67
DiRTRally2.0_60FPS	1	0.5	0.67
DiRTRally2.0_90FPS	0.6	1	0.75
DiRTRally2.0_120FPS	0	0	0
Hellblade_60FPS	1	0.5	0.67
Hellblade_90FPS	0	0	0
Hellblade_120FPS	0.67	1	0.8
RealityMixer+SteamVRHome_60FPS	0.67	1	0.8
RealityMixer+SteamVRHome_90FPS	0.89	0.73	0.8
RealityMixer+SteamVRHome_120FPS	0.75	0.6	0.67
SolarSystemAR_60FPS	0.2	0.33	0.25
SolarSystemAR_90FPS	0	0	0
SolarSystemAR_120FPS	1	1	1
TheLab_60FPS	0.73	0.73	0.73
TheLab_90FPS	0.67	1	0.8
TheLab_120FPS	0.67	0.8	0.73
VRChat_60FPS	1	0.75	0.86
VRChat_90FPS	0	0	0
VRChat_120FPS	0	0	0

Table 16
Classification Performance Summary of Streaming VLS Simulation Datasets at the Flow Level

Resampling	Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
/	1D-CNN	78	0.74	0.73	0.73	0.73	0.73	0.78	0.77
/	RNN	78	0.68	0.67	0.66	0.67	0.67	0.69	0.69
/	GCN	78	0.59	0.58	0.55	0.58	0.58	0.58	0.58
/	Stacking	78	0.90	0.90	0.89	0.90	0.90	0.98	0.90
/	Deep Packet	78	0.68	0.68	0.67	0.68	0.67	0.68	0.68
/	FlowPic	78	0.65	0.64	0.63	0.64	0.64	0.65	0.65
/	FS-Net	78	0.72	0.71	0.71	0.71	0.71	0.72	0.71
SMOTEENN	1D-CNN	78	0.91	0.90	0.87	0.90	0.90	0.92	0.92
SMOTEENN	RNN	78	0.84	0.84	0.78	0.84	0.84	0.84	0.84
SMOTEENN	GCN	78	0.70	0.64	0.57	0.64	0.64	0.69	0.69
SMOTEENN	Stacking	78	0.98	0.98	0.97	0.98	0.98	1.00	0.98
SMOTEENN	Deep Packet	78	0.82	0.82	0.75	0.82	0.82	0.82	0.83
SMOTEENN	FlowPic	78	0.79	0.78	0.71	0.78	0.78	0.78	0.79
SMOTEENN	FS-Net	78	0.87	0.86	0.81	0.86	0.86	0.86	0.86

Table 17
Classification Performance in Each Category of Streaming VLS Simulation Datasets under the Stacking Model after Resampling at the Flow Level

Category	P.	R.	F.	Category	P.	R.	F.
144P30FPS_DASH_VLS_Download	0.98	0.97	0.98	720P60FPS_RTMP_VLS_Download	1	1	1
144P30FPS_HLS_VLS_Download	0.9	0.9	0.9	720P60FPS_RTMP_VLS_Upload	1	1	1
144P30FPS_HTTP_FLV_VLS_Download	1	1	1	720P60FPS_WebRTC_VLS_Download	0.99	1	1
144P30FPS_RTMP_VLS_Download	0.99	1	1	1080P30FPS_DASH_VLS_Download	0.97	0.95	0.96
144P30FPS_RTMP_VLS_Upload	1	1	1	1080P30FPS_HLS_VLS_Download	0.95	0.98	0.97
144P30FPS_WebRTC_VLS_Download	1	1	1	1080P30FPS_HTTP_FLV_VLS_Download	1	1	1
240P30FPS_DASH_VLS_Download	0.97	0.97	0.97	1080P30FPS_RTMP_VLS_Download	0.99	1	1
240P30FPS_HLS_VLS_Download	0.89	0.85	0.87	1080P30FPS_RTMP_VLS_Upload	1	1	1
240P30FPS_HTTP_FLV_VLS_Download	1	1	1	1080P30FPS_WebRTC_VLS_Download	0.91	0.92	0.91
240P30FPS_RTMP_VLS_Download	1	0.99	1	1080P60FPS_DASH_VLS_Download	0.92	0.95	0.93
240P30FPS_RTMP_VLS_Upload	1	1	1	1080P60FPS_HLS_VLS_Download	0.88	0.84	0.86
240P30FPS_WebRTC_VLS_Download	1	0.99	0.99	1080P60FPS_HTTP_FLV_VLS_Download	1	1	1
360P30FPS_DASH_VLS_Download	1	0.98	0.99	1080P60FPS_RTMP_VLS_Download	1	0.99	0.99
360P30FPS_HLS_VLS_Download	0.85	0.88	0.87	1080P60FPS_RTMP_VLS_Upload	1	1	1
360P30FPS_HTTP_FLV_VLS_Download	1	1	1	1080P60FPS_WebRTC_VLS_Download	0.92	0.85	0.88
360P30FPS_RTMP_VLS_Download	1	1	1	1440P30FPS_DASH_VLS_Download	0.95	0.93	0.94
360P30FPS_RTMP_VLS_Upload	1	1	1	1440P30FPS_HLS_VLS_Download	0.81	0.86	0.84
360P30FPS_WebRTC_VLS_Download	0.98	0.99	0.98	1440P30FPS_HTTP_FLV_VLS_Download	1	1	1
480P30FPS_DASH_VLS_Download	0.98	0.97	0.98	1440P30FPS_RTMP_VLS_Download	1	1	1
480P30FPS_HLS_VLS_Download	0.89	0.89	0.89	1440P30FPS_RTMP_VLS_Upload	1	1	1
480P30FPS_HTTP_FLV_VLS_Download	1	1	1	1440P30FPS_WebRTC_VLS_Download	0.93	0.93	0.93
480P30FPS_RTMP_VLS_Download	1	0.99	0.99	1440P60FPS_DASH_VLS_Download	0.94	0.95	0.94
480P30FPS_RTMP_VLS_Upload	1	1	1	1440P60FPS_HLS_VLS_Download	0.82	0.88	0.85
480P30FPS_WebRTC_VLS_Download	1	1	1	1440P60FPS_HTTP_FLV_VLS_Download	1	1	1
540P30FPS_DASH_VLS_Download	0.96	0.98	0.97	1440P60FPS_RTMP_VLS_Download	1	1	1
540P30FPS_HLS_VLS_Download	0.98	0.96	0.97	1440P60FPS_RTMP_VLS_Upload	1	1	1
540P30FPS_HTTP_FLV_VLS_Download	1	1	1	1440P60FPS_WebRTC_VLS_Download	0.9	0.93	0.91
540P30FPS_RTMP_VLS_Download	1	1	1	2160P30FPS_DASH_VLS_Download	0.96	0.98	0.97
540P30FPS_RTMP_VLS_Upload	1	1	1	2160P30FPS_HLS_VLS_Download	0.88	0.82	0.85
540P30FPS_WebRTC_VLS_Download	0.92	0.97	0.94	2160P30FPS_HTTP_FLV_VLS_Download	1	1	1
720P30FPS_DASH_VLS_Download	0.98	0.96	0.97	2160P30FPS_RTMP_VLS_Download	1	0.99	1
720P30FPS_HLS_VLS_Download	1	1	1	2160P30FPS_RTMP_VLS_Upload	1	1	1
720P30FPS_HTTP_FLV_VLS_Download	1	1	1	2160P30FPS_WebRTC_VLS_Download	0.93	0.9	0.91
720P30FPS_RTMP_VLS_Download	0.99	1	1	2160P60FPS_DASH_VLS_Download	0.98	0.99	0.99
720P30FPS_RTMP_VLS_Upload	1	1	1	2160P60FPS_HLS_VLS_Download	0.9	0.87	0.89
720P30FPS_WebRTC_VLS_Download	0.99	0.98	0.98	2160P60FPS_HTTP_FLV_VLS_Download	1	1	1
720P60FPS_DASH_VLS_Download	0.96	0.97	0.97	2160P60FPS_RTMP_VLS_Download	1	1	1
720P60FPS_HLS_VLS_Download	1	1	1	2160P60FPS_RTMP_VLS_Upload	1	1	1
720P60FPS_HTTP-FLV_VLS_Download	1	1	1	2160P60FPS_WebRTC_VLS_Download	1	1	1

3.3. Payload-Level Classification Performance

In the payload-level classification task, all byte payload samples have been converted into graph-based representations. As a result, SMOTEENN resampling cannot be applied because both SMOTE and ENN are designed for tabular datasets. These methods assume that each sample is a fixed-length vector and that distances between samples are meaningful, typically using Euclidean distance. In graph data, where each sample has a unique structure composed of nodes and edges, interpolation between samples is not straightforward. Therefore, techniques like SMOTE and ENN are not applicable to graph-based datasets. In all interactive media network traffic datasets at the payload level, even the 144P_30FPS_DASH_VLS_Download category with the smallest number of PCAP packets, still contains more than 5000 packets. This ensures that there is a sufficient volume of payload-level samples; synthetic augmentation is not required. All eight classification models were evaluated on the payload-level task. With the exception of the 2D-CNN model, which operates directly on the original graph structures, all other models convert the graphs into one-dimensional feature vectors and apply PCA for dimensionality reduction. As shown in Table 18, the 2D-CNN model achieved the highest classification accuracy of 0.81 on the test set. However, both the training and validation accuracies were 0.98, suggesting overfitting. The model memorizes the training and validation data instead of learning patterns that generalize well. The issue is more evident in the 1D-CNN model, where the gap between training and test accuracy exceeds 28 percent.

The other models show similar macro F1-scores, generally around 0.7. The 2D-CNN model had a macro F1-score of approximately 0.82, which is about 15 percent higher than the others. Although the dataset does not suffer from class imbalance, a major challenge is the selection of appropriate feature fusion techniques. Ineffective fusion strategies prevent the model from learning useful classification patterns or defining clear decision boundaries. Table 19 provides a breakdown of performance by category. For example, the 480P_30FPS_VLS_Facebook category had an F1-score of only 0.26, and the 720P_30FPS_VOD_Facebook category had an even lower score of 0.14. In both cases, the feature fusion methods included random pixel scrambling and random pixel shifting, which did not improve performance. Similarly, the 360P_30FPS_VLS_TikTok and 540P_30FPS_VLS_TikTok categories had F1-scores below 0.4. These results were obtained using fusion techniques such as applying random Gaussian blur, alternating small-kernel sharpening, two-dimensional image filtering, and adding random tile patterns. More than 20 categories had F1-scores below 0.80. This indicates that using stochastic noise or small graph alterations through permutations and combinations provides only limited benefit for improving classification performance.

For the Metaverse dataset, the overall classification performance at the payload level is significantly better than the flow-level results, as shown in Table 20 when compared to Table 14. The 2D-CNN model achieved a classification accuracy as high as 0.81, and the macro F1-score reached 0.82. However, there are signs of overfitting, as the training and validation accuracy are notably higher than the test performance. Despite this, the 2D-CNN still outperformed all other models by more than 10 percent in overall classification accuracy. According to Table 21, which presents classification performance by category, only four categories, including

RealityMixer+SteamVRHome_90FPS, SolarSystemAR_60FPS, TheLab_60FPS, and VRChat_60FPS, had F1-scores below 0.6. The remaining categories all achieved F1-scores above 0.8, indicating relatively strong performance across most of the dataset. Several factors contribute to the classification challenges in the lower-performing categories. The original traffic capture was limited to a bandwidth of 15 Mbps and used the WebRTC streaming protocol over TCP. Due to TSO or GSO being enabled, many TCP segments stored in the PCAP files were unfragmented and large, with some payloads exceeding 10,000 bytes. However, only the first 1250 bytes of each packet were used to generate graph samples, and the remaining payload data was omitted. Additionally, a significant number of packets had payloads smaller than 100 bytes. Although the dataset contained a sufficient number of samples, the individual graph samples often lacked informative or effective features, which made accurate classification more difficult. Moreover, many categories relied on similar feature fusion methods, such as random rotation, vertical flipping, Gaussian noise, and Gaussian blur with a random kernel size. These standard augmentation techniques were not able to clearly enhance classification-relevant features, primarily because the original features themselves were limited in quality and informativeness.

The Streaming VLS Simulation dataset shows relatively low classification performance at the payload level. As reported in Table 22, the highest overall classification accuracy reaches only 0.75 under the 2D-CNN model across 78 categories. This is over 20 % lower than the time-series classification accuracy, where the 1D-CNN model achieves 0.91, and the Stacking model reaches 0.98 at the flow level. Analysis of the raw PCAP files highlights protocol-related differences that affect classification performance. WebRTC, which is based on UDP, contains over 80 percent of packets with payloads smaller than 500 bytes. This results in a lack of effective, high-quality samples for graph construction. By contrast, protocols such as DASH, HLS, HTTP-FLV, and RTMP operate over TCP. These protocols often include very large payloads, with some segments exceeding 10,000 bytes and reaching up to 65,549 bytes in the DASH protocol. However, due to storage limitations (1250 bytes) during graph conversion, only approximately 2 percent of the payload bytes are retained, eliminating most of the useful information. Additionally, more than 0.5 percent of packets have payloads smaller than 100 bytes. As shown in Table 23, most DASH-related categories have F1-scores below 0.5. These issues stem from TCP offloading mechanisms, where the operating system and network interface card aggregate large TCP segments for efficiency before splitting them into smaller packets during transmission. When captured on the sender side, especially on loopback interfaces, these large segments appear in the PCAP files even though they are not transmitted as such on the network. In contrast, UDP-based protocols such as WebRTC do not support these offload mechanisms and therefore do not exhibit similar behavior. In the Metaverse dataset, WebRTC is sometimes configured to fall back to TCP due to network restrictions. This fallback leads to similar large TCP segments when TSO or GSO is enabled. Although WebRTC is fundamentally UDP-based, real-world deployments often rely on TCP-based fallback paths, which result in the same offloading effects observed in DASH and HLS traffic. When using 2000 randomly selected samples, the limited payload information in small packets continues to obstruct correct classification. Even after applying graph-based feature fusion techniques, the

results remain poor due to the insufficient quality of the original graph samples. Compared to the Live Streaming dataset, the VLS Simulation dataset includes more TCP-based traffic and a higher proportion of low-quality byte payload samples.

In summary, the classification performance at the payload level varies significantly across interactive media applications. The lack of consistent structural characteristics between traffic types makes it difficult to generalize models. Although a variety of feature fusion techniques were applied to extract common patterns, the limited information in many graph samples reduced their effectiveness. During dataset generation, automatic zero-padding was used to fill insufficient payloads, which contributed little useful information to the graphs. As a result, standard augmentation methods such as shape rotation and the addition of random noise were not effective in improving classification accuracy. These methods are not well-suited to the structure of the data, and more advanced feature fusion strategies need to be explored.

Table 18
Classification Performance Summary of Live Streaming Datasets at the Payload Level

Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
2D-CNN	95	0.82	0.81	0.82	0.81	0.81	0.98	0.98
1D-CNN	95	0.71	0.71	0.71	0.71	0.71	0.99	0.99
RNN	95	0.73	0.73	0.72	0.73	0.73	0.77	0.78
GCN	95	0.70	0.69	0.68	0.69	0.69	0.70	0.68
Stacking	95	0.74	0.74	0.74	0.74	0.74	0.99	0.74
Deep Packet	95	0.73	0.72	0.72	0.72	0.72	0.79	0.80
FlowPic	95	0.72	0.71	0.70	0.71	0.71	0.73	0.73
FS-Net	95	0.75	0.74	0.74	0.74	0.74	0.77	0.77

Table 19
Classification Performance in Each Category of Live Streaming Datasets under the 2D-CNN Model at the Payload Level

Category	P.	R.	F.	Category	P.	R.	F.
480P_3Mbps_N3_10Mhz	0.83	0.69	0.75	VPN_Vimeo	0.99	0.98	0.98
480P_3Mbps_N78_10Mhz	1	1	1	VPN_YouTube	0.52	0.52	0.52
480P_3Mbps_N78_20Mhz	1	1	1	160P_30FPS_VoD_Twitch	0.99	0.98	0.99
720P_6Mbps_N3_10Mhz	1	1	1	160P_30FPS_VLS_Twitch	1	1	1
720P_6Mbps_N78_10Mhz	1	1	1	360P_30FPS_VoD_Twitch	0.48	0.47	0.48
720P_6Mbps_N78_20Mhz	1	1	1	360P_30FPS_VLS_Twitch	0.99	1	0.99
1080P_10Mbps_N3_10Mhz	1	0.98	0.99	480P_30FPS_VoD_Twitch	0.98	0.96	0.97
1080P_10Mbps_N78_10Mhz	1	1	1	480P_30FPS_VLS_Twitch	0.83	0.94	0.88
1080P_10Mbps_N78_20Mhz	0.95	0.99	0.97	720P_30FPS_VoD_Twitch	1	1	1
2160P_20Mbps_N3_10Mhz	0.79	0.92	0.85	720P_60FPS_VoD_Twitch	1	0.97	0.98
2160P_20Mbps_N78_20Mhz	1	0.99	0.99	720P_30FPS_VLS_Twitch	0.67	0.84	0.74
4320P_40Mbps_N3_10Mhz	0.99	0.99	0.99	720P_60FPS_VLS_Twitch	0.72	0.66	0.69
4320P_40Mbps_N78_20Mhz	0.99	0.99	0.99	1080P_30FPS_VoD_Twitch	0.95	1	0.98
2160P_30FPS_VoD_BiliBili	1	0.99	0.99	1080P_60FPS_VoD_Twitch	1	0.99	1
2160P_60FPS_VoD_BiliBili	1	1	1	1080P_30FPS_VLS_Twitch	0.92	0.79	0.85
2160P_120FPS_VoD_BiliBili	0.85	0.85	0.85	1080P_60FPS_VLS_Twitch	0.99	0.96	0.97
2160P_60FPS_VLS_BiliBili	0.69	0.67	0.68	240P_30FPS_VoD_Vimeo	0.74	0.8	0.77
2160P_80FPS_VLS_BiliBili	0.95	0.96	0.96	360P_30FPS_VoD_Vimeo	1	1	1
4320P_30FPS_VoD_BiliBili	0.87	0.87	0.87	540P_30FPS_VoD_Vimeo	0.49	0.49	0.49
360P_30FPS_VoD_BiliBili	0.9	0.89	0.89	720P_30FPS_VoD_Vimeo	0.34	0.38	0.36
480P_30FPS_VoD_BiliBili	0.95	0.87	0.91	1080P_30FPS_VoD_Vimeo	0.65	0.66	0.66
480P_30FPS_VLS_BiliBili	1	1	1	1440P_30FPS_VoD_YouTube	0.5	0.48	0.49
720P_30FPS_VoD_BiliBili	0.55	0.47	0.51	1440P_60FPS_VoD_YouTube	0.41	0.31	0.35
720P_30FPS_VLS_BiliBili	1	0.97	0.98	1440P_30FPS_VLS_YouTube	0.99	1	0.99
1080P_30FPS_VoD_BiliBili	1	1	1	1440P_60FPS_VLS_YouTube	0.26	0.29	0.27
1080P_60FPS_VoD_BiliBili	0.99	1	1	2160P_30FPS_VoD_YouTube	0.27	0.43	0.33
1080P_60FPS_VLS_BiliBili	0.89	0.94	0.92	2160P_60FPS_VoD_YouTube	0.81	0.68	0.74
1440P_60FPS_VLS_BiliBili	1	0.99	0.99	2160P_30FPS_VLS_YouTube	0.97	0.99	0.98

Table 19 (continued)

144P_30FPS_VOD_Facebook	0.98	1	0.99	2160P_60FPS_VLS_YouTube	0.99	0.99	0.99
144P_30FPS_VLS_Facebook	0.8	0.72	0.76	4320P_30FPS_VoD_YouTube	0.99	1	0.99
240P_30FPS_VLS_Facebook	0.97	0.99	0.98	4320P_60FPS_VoD_YouTube	0.96	0.94	0.95
360P_30FPS_VOD_Facebook	0.98	0.91	0.94	144P_30FPS_VoD_YouTube	0.78	1	0.88
360P_30FPS_VLS_Facebook	1	1	1	144P_30FPS_VLS_YouTube	0.97	1	0.98
480P_30FPS_VOD_Facebook	1	1	1	240P_30FPS_VoD_YouTube	1	1	1
480P_30FPS_VLS_Facebook	0.27	0.26	0.26	240P_30FPS_VLS_YouTube	1	1	1
540P_30FPS_VOD_Facebook	0.62	0.65	0.64	360P_30FPS_VoD_YouTube	1	0.89	0.94
720P_30FPS_VOD_Facebook	0.17	0.12	0.14	360P_30FPS_VLS_YouTube	0.96	0.86	0.91
720P_30FPS_VLS_Facebook	0.9	0.76	0.82	480P_30FPS_VoD_YouTube	0.83	0.75	0.79
1080P_30FPS_VOD_Facebook	0.77	0.87	0.82	480P_30FPS_VLS_YouTube	0.51	0.46	0.48
1080P_30FPS_VLS_Facebook	0.99	0.99	0.99	720P_30FPS_VoD_YouTube	0.34	0.4	0.37
360P_30FPS_VLS_TikTok	0.3	0.39	0.34	720P_60FPS_VoD_YouTube	1	0.99	1
540P_30FPS_VLS_TikTok	0.4	0.37	0.38	720P_30FPS_VLS_YouTube	0.99	0.99	0.99
720P_30FPS_VLS_TikTok	0.85	1	0.92	720P_60FPS_VLS_YouTube	0.72	0.63	0.67
720P_60FPS_VLS_TikTok	0.56	0.64	0.6	1080P_30FPS_VoD_YouTube	0.51	0.47	0.49
1080P_30FPS_VLS_TikTok	0.31	0.2	0.24	1080P_60FPS_VoD_YouTube	1	1	1
1080P_60FPS_VLS_TikTok	0.62	0.58	0.6	1080P_30FPS_VLS_YouTube	1	1	1
Tor_Vimeo	0.62	0.6	0.61	1080P_60FPS_VLS_YouTube	0.23	0.3	0.26
Tor_YouTube	1	0.9	0.95				

Table 20

Classification Performance Summary of Metaverse Datasets at the Payload Level

Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
2D-CNN	21	0.84	0.84	0.85	0.84	0.84	1.00	1.00
1D-CNN	21	0.79	0.79	0.79	0.79	0.79	1.00	0.99
RNN	21	0.79	0.78	0.79	0.78	0.78	0.89	0.89
GCN	21	0.73	0.73	0.73	0.73	0.73	0.78	0.72
Stacking	21	0.82	0.82	0.82	0.82	0.82	1.00	0.82
Deep Packet	21	0.79	0.79	0.80	0.79	0.79	0.95	0.95
FlowPic	21	0.78	0.78	0.78	0.78	0.78	0.81	0.81
FS-Net	21	0.81	0.81	0.81	0.81	0.80	0.87	0.87

Table 21

Classification Performance in Each Category of Metaverse Datasets under the 2D-CNN Model at the Payload Level

Category	P.	R.	F.
BigscreenTheatre_60FPS	0.77	0.84	0.8
BigscreenTheatre_90FPS	1	1	1
BigscreenTheatre_120FPS	1	1	1
DiRTRally2.0_60FPS	0.98	0.99	0.98
DiRTRally2.0_90FPS	0.81	0.82	0.82
DiRTRally2.0_120FPS	0.93	0.94	0.93
Hellblade_60FPS	0.81	0.87	0.84
Hellblade_90FPS	0.99	1	0.99
Hellblade_120FPS	0.94	0.9	0.92
RealityMixer+SteamVRHome_60FPS	0.84	0.93	0.88
RealityMixer+SteamVRHome_90FPS	0.54	0.55	0.54
RealityMixer+SteamVRHome_120FPS	0.89	0.85	0.87
SolarSystemAR_60FPS	0.52	0.6	0.56
SolarSystemAR_90FPS	0.96	0.92	0.94
SolarSystemAR_120FPS	0.9	0.85	0.87
TheLab_60FPS	0.66	0.63	0.64
TheLab_90FPS	0.92	0.8	0.86
TheLab_120FPS	0.82	0.8	0.81
VRChat_60FPS	0.61	0.64	0.62
VRChat_90FPS	0.88	0.74	0.8
VRChat_120FPS	1	0.99	1

Table 22
Classification Performance Summary of Streaming VLS Simulation Datasets at the Payload Level

Model	Category	P.W	R.W	F.A	F.I	A.Test	A.Train	A.Validate
2D-CNN	78	0.76	0.75	0.75	0.75	0.75	0.99	0.99
1D-CNN	78	0.68	0.67	0.67	0.67	0.67	0.98	0.98
RNN	78	0.68	0.65	0.65	0.65	0.65	0.71	0.71
GCN	78	0.60	0.57	0.57	0.57	0.57	0.61	0.56
Stacking	78	0.65	0.65	0.65	0.65	0.65	0.99	0.65
Deep Packet	78	0.64	0.63	0.63	0.63	0.64	0.77	0.77
FlowPic	78	0.65	0.63	0.62	0.63	0.63	0.65	0.65
FS-Net	78	0.67	0.66	0.66	0.66	0.66	0.73	0.73

Table 23
Classification Performance in Each Category of Streaming VLS Simulation Datasets under the 2D-CNN Model at the Payload Level

Category	P.	R.	F.	Category	P.	R.	F.
144P30FPS_DASH_VLS_Download	0.99	0.94	0.96	720P60FPS_RTMP_VLS_Download	0.55	0.63	0.59
144P30FPS_HLS_VLS_Download	1	1	1	720P60FPS_RTMP_VLS_Upload	0.66	0.64	0.65
144P30FPS_HTTP_FLV_VLS_Download	1	1	1	720P60FPS_WebRTC_VLS_Download	0.87	0.95	0.91
144P30FPS_RTMP_VLS_Download	0.99	1	1	1080P30FPS_DASH_VLS_Download	0.11	0.15	0.13
144P30FPS_RTMP_VLS_Upload	0.94	0.88	0.91	1080P30FPS_HLS_VLS_Download	0.76	0.63	0.69
144P30FPS_WebRTC_VLS_Download	1	1	1	1080P30FPS_HTTP_FLV_VLS_Download	0.61	0.73	0.67
240P30FPS_DASH_VLS_Download	0.96	0.9	0.93	1080P30FPS_RTMP_VLS_Download	0.78	0.73	0.76
240P30FPS_HLS_VLS_Download	0.99	0.96	0.98	1080P30FPS_RTMP_VLS_Upload	0.53	0.57	0.55
240P30FPS_HTTP_FLV_VLS_Download	0.71	0.77	0.74	1080P30FPS_WebRTC_VLS_Download	0.99	0.99	0.99
240P30FPS_RTMP_VLS_Download	0.91	0.83	0.87	1080P60FPS_DASH_VLS_Download	0.41	0.33	0.37
240P30FPS_RTMP_VLS_Upload	0.61	0.67	0.64	1080P60FPS_HLS_VLS_Download	0.85	0.65	0.74
240P30FPS_WebRTC_VLS_Download	0.99	1	1	1080P60FPS_HTTP_FLV_VLS_Download	1	0.88	0.94
360P30FPS_DASH_VLS_Download	0.92	0.81	0.86	1080P60FPS_RTMP_VLS_Download	1	1	1
360P30FPS_HLS_VLS_Download	0.99	0.91	0.95	1080P60FPS_RTMP_VLS_Upload	0.76	0.77	0.76
360P30FPS_HTTP_FLV_VLS_Download	0.52	0.59	0.55	1080P60FPS_WebRTC_VLS_Download	1	0.99	0.99
360P30FPS_RTMP_VLS_Download	0.52	0.61	0.56	1440P30FPS_DASH_VLS_Download	0.83	0.66	0.74
360P30FPS_RTMP_VLS_Upload	0.56	0.53	0.54	1440P30FPS_HLS_VLS_Download	0.91	0.89	0.9
360P30FPS_WebRTC_VLS_Download	0.96	0.94	0.95	1440P30FPS_HTTP_FLV_VLS_Download	0.94	0.99	0.96
480P30FPS_DASH_VLS_Download	0.87	0.82	0.84	1440P30FPS_RTMP_VLS_Download	0.18	0.34	0.24
480P30FPS_HLS_VLS_Download	0.85	0.75	0.8	1440P30FPS_RTMP_VLS_Upload	0.33	0.45	0.38
480P30FPS_HTTP_FLV_VLS_Download	0.86	0.91	0.89	1440P30FPS_WebRTC_VLS_Download	1	0.99	0.99
480P30FPS_RTMP_VLS_Download	1	0.97	0.98	1440P60FPS_DASH_VLS_Download	0.23	0.19	0.21
480P30FPS_RTMP_VLS_Upload	0.88	0.54	0.67	1440P60FPS_HLS_VLS_Download	0.68	0.69	0.69
480P30FPS_WebRTC_VLS_Download	1	1	1	1440P60FPS_HTTP_FLV_VLS_Download	0.17	0.17	0.17
540P30FPS_DASH_VLS_Download	0.99	1	0.99	1440P60FPS_RTMP_VLS_Download	0.85	0.89	0.87
540P30FPS_HLS_VLS_Download	0.85	0.71	0.78	1440P60FPS_RTMP_VLS_Upload	0.81	0.78	0.8
540P30FPS_HTTP_FLV_VLS_Download	0.8	0.76	0.78	1440P60FPS_WebRTC_VLS_Download	1	1	1
540P30FPS_RTMP_VLS_Download	0.77	0.86	0.81	2160P30FPS_DASH_VLS_Download	0.56	0.66	0.61
540P30FPS_RTMP_VLS_Upload	0.98	0.9	0.94	2160P30FPS_HLS_VLS_Download	0.72	0.62	0.67
540P30FPS_WebRTC_VLS_Download	0.92	0.89	0.9	2160P30FPS_HTTP_FLV_VLS_Download	0.76	0.82	0.79
720P30FPS_DASH_VLS_Download	0.65	0.99	0.79	2160P30FPS_RTMP_VLS_Download	0.8	0.79	0.8
720P30FPS_HLS_VLS_Download	0.37	0.45	0.41	2160P30FPS_RTMP_VLS_Upload	0.59	0.53	0.56
720P30FPS_HTTP_FLV_VLS_Download	1	1	1	2160P30FPS_WebRTC_VLS_Download	1	1	1
720P30FPS_RTMP_VLS_Download	0.99	0.99	0.99	2160P60FPS_DASH_VLS_Download	0.14	0.16	0.15
720P30FPS_RTMP_VLS_Upload	0.36	0.29	0.32	2160P60FPS_HLS_VLS_Download	0.79	0.69	0.74
720P30FPS_WebRTC_VLS_Download	0.97	0.93	0.95	2160P60FPS_HTTP_FLV_VLS_Download	0.19	0.16	0.17
720P60FPS_DASH_VLS_Download	0.41	0.36	0.39	2160P60FPS_RTMP_VLS_Download	0.11	0.1	0.11
720P60FPS_HLS_VLS_Download	0.85	0.72	0.78	2160P60FPS_RTMP_VLS_Upload	0.82	0.73	0.77
720P60FPS_HTTP-FLV_VLS_Download	1	0.95	0.97	2160P60FPS_WebRTC_VLS_Download	1	1	1

4. CONCLUSIONS

The Doctoral Thesis presents a comprehensive approach to achieve fine-grained classification of interactive media applications network traffic across multiple granularities, including the time series, flow, and payload levels. The main achievements are summarized as follows.

1. The scope of traditional video category network traffic, which is typically limited to general Internet applications and protocol-specific datasets, has been significantly expanded. This study redefines the classification scope to encompass three representative and widely used types of interactive media applications: OTT Live Streaming, Cloud Gaming, and Metaverse applications. These categories share common features, such as high traffic volume between clients and servers, and the presence of continuous bidirectional interaction over diverse and complex network environments. The resulting traffic exhibits unique and distinguishable patterns not captured by traditional coarse-grained video classification approaches. Unlike conventional classification tasks that rely solely on simple application names, service identifiers, or protocol labels as ground-truth, this study emphasizes the need for deeper context. Conventional methods often lack clarity in the relationship between the ground-truth labels and actual traffic characteristics. Furthermore, existing research rarely provides a comprehensive analysis of the factors influencing video traffic patterns.

This study systematically investigates the impact of various video-related factors on interactive media network traffic. These include video quality parameters such as resolution, frame rate, compression format, container type, bitrate, motion complexity, content origin, and hardware capabilities of both client and server devices. In addition, it considers transmission parameters like segment duration, GOP structure, codec type, encoding preset, maximum bitrate, buffer size, and transport protocols, including TCP, UDP, and QUIC. Application-level streaming protocols examined include DASH, HLS, RTMP, RTCP, RTP, SRT, HTTP-FLV, and WebRTC. Network condition variables such as 4G, 5G, Wi-Fi, available bandwidth, QoS, QoE, and the usage of VPNs or Tor for encryption and anonymization are also considered. These factors were analyzed to determine their influence on the characteristics of interactive media network traffic. Based on the findings, traffic categories in the classification tasks were defined using combinations of the most impactful factors, such as platform, application name, resolution, frame rate, and traffic direction (uplink and downlink). This approach enabled the construction of highly detailed, fine-grained interactive media network traffic datasets. A total of over 84 GB of traffic data was collected, covering both real-world applications and idealized testing environments. The datasets include both public and proprietary sources, ensuring a diverse and representative sample of interactive media network traffic.

2. The fine-grained interactive media network traffic datasets developed in the Doctoral Thesis are characterized not only by a wide range of category diversity but also by multiple levels of classification granularity, including time series level, flow level, and payload level. Each level addresses specific use-case scenarios. When raw traffic PCAP files are unavailable,

time series datasets constructed from packets, bytes and timestamps provide valuable alternatives. These temporal representations enable the rapid identification of encrypted traffic based on timing patterns, without requiring access to raw PCAP contents. At the flow level, samples capture general traffic patterns, which may manifest as elephant or mouse-specific flows, which are particularly useful for identifying distinct behavioral characteristics of interactive media applications. These flow-level features, combined with partial DPI, contribute to more accurate classification of application types and protocols. At the payload level, classification is especially advantageous in situations with limited PCAP data capture durations or sparse session streams. Due to the bursty nature of interactive media network traffic, even brief capture periods typically yield sufficient byte-level samples. These payload samples, when transformed into graph representations, enable the extraction of both spatial and temporal features, offering a richer structure than one-dimensional data formats. The datasets constructed at all three granularities exhibit transparent, explainable preprocessing procedures and ensure reliability and reproducibility of the samples.

3. This study introduces a multi-layer 1D-CNN tailored for fine-grained traffic classification. Building upon this architecture, a modified 2D-CNN model was developed specifically for graph-based classification, along with an RNN and a GCN designed for flow-level tasks. Across all three granularities, the proposed 1D-CNN consistently outperforms existing state-of-the-art models, including Deep Packet, FlowPic, and FS-Net, demonstrating strong adaptability and robustness across diverse traffic scenarios. A Stacking model, composed of multiple meta-learners, was also implemented and found to be particularly effective at the flow level. Its ability to handle high-variance features and large sample volumes resulted in classification accuracy exceeding 1D-CNN by more than 10 % under the same conditions. At the payload level, the 2D-CNN model achieved the best performance for graph-based classification. Its advantage stems from its capacity to leverage two-dimensional spatiotemporal features, surpassing one-dimensional models by at least 10 % in overall classification accuracy. In summary, model selection is highly dependent on dataset characteristics, feature construction, and scenario-specific requirements. For the fine-grained classification of interactive media network traffic, all proposed models achieved strong and consistent performance across their respective domains.

Future research can be summarized as follows.

1. At the time series-level and flow-level classification stages, a persistent issue is the imbalance in the volume of samples across different traffic categories. While this can be mitigated by actively capturing more raw traffic data or passively collecting longer flow data, synthetic data generation remains a common approach. However, the SMOTEENN method used in this work shows limitations when applied to small datasets, as it may only execute the ENN component without generating new synthetic samples. Therefore, more advanced data augmentation techniques need to be explored for better handling of imbalanced datasets.

2. In the time series-level classification task, only packet and byte vectors were used to construct the samples through the proposed universal method in Algorithm 4. Further research includes examining how applying other elements in Algorithm 4 and the corresponding time series distributions affect the classification results. Notably, not all categories demonstrate a clear increase in traffic volume with higher video resolution or frame rate. Some categories show irregular packet bursts and high noise levels in their time series, which negatively impact classification. Applying appropriate filtering methods may help reduce the influence of these noise patterns.

3. For payload-level classification, the current method pads packet payloads smaller than 1250 bytes with zeros and truncates those larger than the threshold. This results in considerable data loss, especially for large payloads associated with TCP-based streaming protocols using features like TSO. A potential improvement involves reducing the grayscale graph size from 100×100 to 32×32, corresponding to 128 bytes. For payloads exceeding this size, additional graphs can be generated to retain all relevant information. Furthermore, existing graph feature fusion methods have shown limited effectiveness in improving fine-grained classification accuracy. More advanced fusion strategies are needed to better leverage structural and contextual information in payload samples.

4. Both one-dimensional temporal and two-dimensional spatiotemporal features used in training can be standardized and stored as pre-trained representations to enable online, real-time classification of interactive media traffic. Beyond classification, these features may also be valuable for tasks such as traffic prediction, anomaly detection, and identifying malicious activity disguised as normal traffic. These directions offer practical value and merit further investigation. In addition, interactive media network traffic patterns after training can be used to identify anonymized and encrypted traffic. This is particularly relevant as generative artificial intelligence (AI) applications such as text-to-video generation, GPU-based deep learning super sampling (DLSS), and motion interpolation for gaming or metaverse environments become more prevalent. These emerging technologies not only improve users' QoE but also contribute to significant increases in traffic volume. In future heterogeneous networks, such traffic patterns may provide valuable prior knowledge to support further research.

BIBLIOGRAPHY

1. A., P. *IDC: Expect 175 Zettabytes of Data Worldwide by 2025*, *Network World*; 2018; Available online: https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html
2. Abbasi, M.; Shahraki, A.; Taherkordi, A. Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey. *Computer Communications* **2021**, *170*, 19–41, doi:10.1016/j.comcom.2021.01.021.
3. Fowdur, T. P.; Babooram, L. *Machine Learning for Network Traffic and Video Quality Analysis: Develop and Deploy Applications Using JavaScript and Node. Js*; Apress: Berkeley, CA, 2024; ISBN 979-8-8688-0353-6.
4. Nguyen, T. T. T.; Armitage, G. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *IEEE Commun. Surv. Tutorials* **2008**, *10*, 56–76, doi:10.1109/SURV.2008.080406.
5. Yan, J.; Yuan, J. A Survey of Traffic Classification in Software-Defined Networks. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN); IEEE: Shenzhen, August 2018; pp. 200–206.
6. Wang, P.; Chen, X.; Ye, F.; Sun, Z. A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning. *IEEE Access* **2019**, *7*, 54024–54033, doi:10.1109/ACCESS.2019.2912896.
7. Tahaei, H.; Afifi, F.; Asemi, A.; Zaki, F.; Anuar, N. B. The Rise of Traffic Classification in IoT Networks: A Survey. *Journal of Network and Computer Applications* **2020**, *154*, 102538, doi:10.1016/j.jnca.2020.102538.
8. Gunavathie; UmaMaheswariS. A Survey on Traffic Prediction and Classification in SDN. In *Advances in Parallel Computing*; Jude Hemanth, D., Ambeth Kumar, V. D., Malathi, S., Eds.; IOS Press, 2020 ISBN 978-1-64368-102-3.
9. Zhao, J.; Jing, X.; Yan, Z.; Pedrycz, W. Network Traffic Classification for Data Fusion: A Survey. *Information Fusion* **2021**, *72*, 22–47, doi:10.1016/j.inffus.2021.02.009.
10. Murshed, M. G. S.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. Machine Learning at the Network Edge: A Survey. *ACM Comput. Surv.* **2022**, *54*, 1–37, doi:10.1145/3469029.
11. Adeleke, O. A.; Bastin, N.; Gurkan, D. Network Traffic Generation: A Survey and Methodology. *ACM Comput. Surv.* **2023**, *55*, 1–23, doi:10.1145/3488375.
12. Azab, A.; Khasawneh, M.; Alrabaee, S.; Choo, K.-K. R.; Sarsour, M. Network Traffic Classification: Techniques, Datasets, and Challenges. *Digital Communications and Networks* **2024**, *10*, 676–692, doi:10.1016/j.dcan.2022.09.009.
13. Boutaba, R.; Salahuddin, M. A.; Limam, N.; Ayoubi, S.; Shahriar, N.; Estrada-Solano, F.; Caicedo, O. M. A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities. *J Internet Serv Appl* **2018**, *9*, 16, doi:10.1186/s13174-018-0087-2.
14. *VNI Complete Forecast Highlights Global Internet Users: % of Population Devices and Connections per Capita Average Speeds Average Traffic per Capita per Month Global: 2021 Forecast Highlights IP Traffic*; 2016; Available online: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf
15. Sandvine *How 'App QoE' Can Increase Profitability While Improving Subscriber Satisfaction*; 2023; Available online: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2023/PDF/BRO-Sandvine-FEB2023.pdf

16. Kwak, K. T.; Lee, S. Y.; Ham, M.; Lee, S. W. The Effects of Internet Proliferation on Search Engine and Over-the-Top Service Markets. *Telecommunications Policy* **2021**, *45*, 102146, doi:10.1016/j.telpol.2021.102146.
17. Che, X.; Ip, B.; Lin, L. A Survey of Current YouTube Video Characteristics. *IEEE MultiMedia* **2015**, *22*, 56–63, doi:10.1109/MMUL.2015.34.
18. Bilal, K.; Erbad, A. Impact of Multiple Video Representations in Live Streaming: A Cost, Bandwidth, and QoE Analysis. In Proceedings of the 2017 IEEE International Conference on Cloud Engineering (IC2E); IEEE: Vancouver, BC, Canada, April 2017; pp. 88–94.
19. Pan, W.; Cheng, G. QoE Assessment of Encrypted YouTube Adaptive Streaming for Energy Saving in Smart Cities. *IEEE Access* **2018**, *6*, 25142–25156, doi:10.1109/ACCESS.2018.2811416.
20. Bentaleb, A.; Taani, B.; Begen, A. C.; Timmerer, C.; Zimmermann, R. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 562–585, doi:10.1109/COMST.2018.2862938.
21. Bermudez, H.-F.; Martinez-Caro, J.-M.; Sanchez-Iborra, R.; Arciniegas, J. L.; Cano, M.-D. Live Video-Streaming Evaluation Using the ITU-T P.1203 QoE Model in LTE Networks. *Computer Networks* **2019**, *165*, 106967, doi:10.1016/j.comnet.2019.106967.
22. Jimenez, L. R.; Solera, M.; Toril, M. A Network-Layer QoE Model for YouTube Live in Wireless Networks. *IEEE Access* **2019**, *7*, 70237–70252, doi:10.1109/ACCESS.2019.2918433.
23. Lee, R.; Venieris, S. I.; Lane, N. D. Deep Neural Network-Based Enhancement for Image and Video Streaming Systems: A Survey and Future Directions. *ACM Comput. Surv.* **2022**, *54*, 1–30, doi:10.1145/3469094.
24. Sultan, M. T.; El Sayed, H. QoE-Aware Analysis and Management of Multimedia Services in 5G and Beyond Heterogeneous Networks. *IEEE Access* **2023**, *11*, 77679–77688, doi:10.1109/ACCESS.2023.3298556.
25. Zhang, Z.; Liu, L.; Lu, X.; Yan, Z.; Li, H. Encrypted Network Traffic Classification: A Data Driven Approach. In Proceedings of the 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom); IEEE: Exeter, United Kingdom, October 2020; p. 7.
26. Zhao, Y.; Yang, Y.; Tian, B.; Yang, J.; Zhang, T.; Hu, N. Edge Intelligence-Based Identification and Classification of Encrypted Traffic of Internet of Things. *IEEE Access* **2021**, *9*, 21895–21903, doi:10.1109/ACCESS.2021.3056216.
27. Canavese, D.; Regano, L.; Basile, C.; Ciravegna, G.; Lioy, A. Data Set and Machine Learning Models for the Classification of Network Traffic Originators. *Data in Brief* **2022**, *41*, 107968, doi:10.1016/j.dib.2022.107968.
28. Bader, O.; Lichy, A.; Dvir, A.; Dubin, R.; Hajaj, C. OSF-EIMTC: An Open-Source Framework for Standardized Encrypted Internet Traffic Classification. *Computer Communications* **2024**, *213*, 271–284, doi:10.1016/j.comcom.2023.10.011.
29. Draper-Gil, G.; Lashkari, A. H.; Mamun, M. S. I.; A. Ghorbani, A. Characterization of Encrypted and VPN Traffic Using Time-Related Features. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy; SCITEPRESS – Science and Technology Publications: Rome, Italy, 2016; pp. 407–414.
30. Habibi Lashkari, A.; Draper Gil, G.; Mamun, M. S. I.; Ghorbani, A. A. Characterization of Tor Traffic Using Time-Based Features. In Proceedings of the 3rd International Conference on Information Systems Security and Privacy; SCITEPRESS – Science and Technology Publications: Porto, Portugal, 2017; pp. 253–262.

31. Aceto, G.; Ciuonzo, D.; Montieri, A.; Persico, V.; Pescape, A. MIRAGE: Mobile-App Traffic Capture and Ground-Truth Creation. In Proceedings of the 2019 4th International Conference on Computing, Communications and Security (ICCCS); IEEE: Rome, Italy, October 2019; pp. 1–8.
32. Habibi Lashkari, A.; Kaur, G.; Rahali, A. DIDarknet: A Contemporary Approach to Detect and Characterize the Darknet Traffic Using Deep Image Learning. In Proceedings of the 2020 10th International Conference on Communication and Network Security; ACM: Tokyo, Japan, November 27, 2020; pp. 1–13.
33. Pham, T.-D.; Ho, T.-L.; Truong-Huu, T.; Cao, T.-D.; Truong, H.-L. MAppGraph: Mobile-App Classification on Encrypted Network Traffic Using Deep Graph Convolution Neural Networks. In Proceedings of the Annual Computer Security Applications Conference; ACM: Virtual Event USA, December 6, 2021; pp. 1025–1038.
34. Dener, M.; Al, S.; Ok, G. RFSE-GRU: Data Balanced Classification Model for Mobile Encrypted Traffic in Big Data Environment. *IEEE Access* **2023**, *11*, 21831–21847, doi:10.1109/ACCESS.2023.3251745.
35. Lan, Y.; Sun, Y.; Liu, S.-P.; Ma, Z.-Z. A Real-Time Network Traffic Analysis and QoS Management Platform. In Proceedings of the 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN); IEEE: Guangzhou, May 2017; pp. 266–270.
36. Orsolic, I.; Suznjevic, M.; Skorin-Kapov, L. YouTube QoE Estimation from Encrypted Traffic: Comparison of Test Methodologies and Machine Learning Based Models. In Proceedings of the 2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX); IEEE: Cagliari, May 2018; pp. 1–6.
37. Zhang, H.; Dong, L.; Gao, G.; Hu, H.; Wen, Y.; Guan, K. DeepQoE: A Multimodal Learning Framework for Video Quality of Experience (QoE) Prediction. *IEEE Trans. Multimedia* **2020**, *22*, 3210–3223, doi:10.1109/TMM.2020.2973828.
38. Wassermann, S.; Seufert, M.; Casas, P.; Gang, L.; Li, K. ViCrypt to the Rescue: Real-Time, Machine-Learning-Driven Video-QoE Monitoring for Encrypted Streaming Traffic. *IEEE Trans. Netw. Serv. Manage.* **2020**, *17*, 2007–2023, doi:10.1109/TNSM.2020.3036497.
39. Huang, Y.-F.; Lin, C.-B.; Chung, C.-M.; Chen, C.-M. Research on QoS Classification of Network Encrypted Traffic Behavior-Based on Machine Learning. *Electronics* **2021**, *10*, 1376, doi:10.3390/electronics10121376.
40. Wu, Z.; Dong, Y.; Qiu, X.; Jin, J. Online Multimedia Traffic Classification from the QoS Perspective Using Deep Learning. *Computer Networks* **2022**, *204*, 108716, doi:10.1016/j.comnet.2021.108716.
41. Al-Fayoumi, M.; Al-Fawa'reh, M.; Nashwan, S. VPN and Non-VPN Network Traffic Classification Using Time-Related Features. *Computers, Materials & Continua* **2022**, *72*, 3091–3111, doi:10.32604/cmc.2022.025103.
42. Li, W.; Canini, M.; Moore, A. W.; Bolla, R. Efficient Application Identification and the Temporal and Spatial Stability of Classification Schema. *Computer Networks* **2009**, *53*, 790–809, doi:10.1016/j.comnet.2008.11.016.
43. Muehlstein, J.; Zion, Y.; Bahumi, M.; Kirshenboim, I.; Dubin, R.; Dvir, A.; Pele, O. Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser and Application. In Proceedings of the 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC); IEEE: Las Vegas, NV, USA, January 2017; pp. 1–6.
44. Wei Wang; Ming Zhu; Xuewen Zeng; Xiaozhou Ye; Yiqiang Sheng Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. In

Proceedings of the 2017 International Conference on Information Networking (ICOIN); IEEE: Da Nang, Vietnam, 2017; pp. 712–717.
45. Arestrom, E.; Carlsson, N. Early Online Classification of Encrypted Traffic Streams Using Multi-Fractal Features. In Proceedings of the IEEE INFOCOM 2019 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); IEEE: Paris, France, April 2019; pp. 84–89.
46. Xiao, X.; Xiao, W.; Li, R.; Luo, X.; Zheng, H.-T.; Xia, S.-T. EBSNN: Extended Byte Segment Neural Network for Network Traffic Classification. *IEEE Trans. Dependable and Secure Comput.* **2021**, 1–1, doi:10.1109/TDSC.2021.3101311.
47. Wu, H.; Wang, L.; Cheng, G.; Hu, X. Mobile Application Encryption Traffic Classification Based On TLS Flow Sequence Network. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops); IEEE: Montreal, QC, Canada, June 2021; pp. 1–6.
48. Shahraki, A.; Abbasi, M.; Taherkordi, A.; Kaosar, M. Internet Traffic Classification Using an Ensemble of Deep Convolutional Neural Networks. In Proceedings of the Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility; ACM: Virtual Event USA, August 23, 2021; pp. 38–43.
49. Grabs, E.; Chen, T.; Petersons, E.; Efrosinin, D.; Ipatovs, A.; Kluga, J.; Culkovs, D. Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks. In Proceedings of the 2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW); IEEE: Riga, Latvia, October 7, 2021; pp. 58–63.
50. Ren, X.; Gu, H.; Wei, W. Tree-RNN: Tree Structural Recurrent Neural Network for Network Traffic Classification. *Expert Systems with Applications* **2021**, *167*, 114363, doi:10.1016/j.eswa.2020.114363.
51. Chen, T.; Grabs, E.; Petersons, E.; Efrosinin, D.; Ipatovs, A.; Kluga, J. Encapsulated and Anonymized Network Video Traffic Classification With Generative Models. In Proceedings of the 2022 Workshop on Microwave Theory and Techniques in Wireless Communications (MTTW); IEEE: Riga, Latvia, October 5, 2022; pp. 13–18.
52. Chen, T.; Grabs, E.; Petersons, E.; Efrosinin, D.; Ipatovs, A.; Bogdanovs, N.; Rjazanovs, D. Multiclass Live Streaming Video Quality Classification Based on Convolutional Neural Networks. *Aut. Control Comp. Sci.* **2022**, *56*, 455–466, doi:10.3103/S0146411622050029.
53. Salman, O.; Elhajj, I. H.; Chehab, A.; Kayssi, A. Towards Efficient Real-Time Traffic Classifier: A Confidence Measure with Ensemble Deep Learning. *Computer Networks* **2022**, *204*, 108684, doi:10.1016/j.comnet.2021.108684.
54. Gabilondo, Á.; Fernández, Z.; Viola, R.; Martín, Á.; Zorrilla, M.; Angueira, P.; Montalbán, J. Traffic Classification for Network Slicing in Mobile Networks. *Electronics* **2022**, *11*, 1097, doi:10.3390/electronics11071097.
55. Luxemburk, J.; Čejka, T. Fine-Grained TLS Services Classification with Reject Option. *Computer Networks* **2023**, *220*, 109467, doi:10.1016/j.comnet.2022.109467.
56. Qin, T.; Cheng, G.; Wei, Y.; Yao, Z. Hier-SFL: Client-Edge-Cloud Collaborative Traffic Classification Framework Based on Hierarchical Federated Split Learning. *Future Generation Computer Systems* **2023**, *149*, 12–24, doi:10.1016/j.future.2023.07.001.
57. Chen, T.; Grabs, E.; Ipatovs, A.; Pētersons, E.; Ancāns, A. Bitrate-Based Video Traffic Classification. In Proceedings of the 2023 Photonics & Electromagnetics Research Symposium (PIERS); IEEE: Prague, Czech Republic, July 3, 2023; pp. 509–514.

58. Zhu, Y.; Tao, J.; Wang, H.; Yu, L.; Luo, Y.; Qi, T.; Wang, Z.; Xu, Y. DGNN: Accurate Darknet Application Classification Adopting Attention Graph Neural Network. *IEEE Trans. Netw. Serv. Manage.* **2024**, *21*, 1660–1671, doi:10.1109/TNSM.2023.3344580.
59. Abolfathi, M.; Inturi, S.; Banaei-Kashani, F.; Jafarian, J. H. Toward Enhancing Web Privacy on HTTPS Traffic: A Novel SuperLearner Attack Model and an Efficient Defense Approach with Adversarial Examples. *Computers & Security* **2024**, *139*, 103673, doi:10.1016/j.cose.2023.103673.
60. Liu, L.; Yu, Y.; Wu, Y.; Hui, Z.; Lin, J.; Hu, J. Method for Multi-Task Learning Fusion Network Traffic Classification to Address Small Sample Labels. *Sci. Rep.* **2024**, *14*, 2518, doi:10.1038/s41598-024-51933-8.
61. Li, Z.; Xu, X. L2-BiTCN-CNN: Spatio-Temporal Features Fusion-Based Multi-Classification Model for Various Internet Applications Identification. *Computer Networks* **2024**, *243*, 110298, doi:10.1016/j.comnet.2024.110298.
62. Wang, C.; Zhang, W.; Hao, H.; Shi, H. Network Traffic Classification Model Based on Spatio-Temporal Feature Extraction. *Electronics* **2024**, *13*, 1236, doi:10.3390/electronics13071236.
63. Park, J.-T.; Shin, C.-Y.; Baek, U.-J.; Kim, M.-S. Fast and Accurate Multi-Task Learning for Encrypted Network Traffic Classification. *Applied Sciences* **2024**, *14*, 3073, doi:10.3390/app14073073.
64. Chen, Z.; Cheng, G.; Wei, Z.; Niu, D.; Fu, N. Classify Traffic Rather Than Flow: Versatile Multi-Flow Encrypted Traffic Classification With Flow Clustering. *IEEE Trans. Netw. Serv. Manage.* **2024**, *21*, 1446–1466, doi:10.1109/TNSM.2023.3322861.
65. Alkadi, S.; Al-Ahmadi, S.; Ben Ismail, M. M. RobEns: Robust Ensemble Adversarial Machine Learning Framework for Securing IoT Traffic. *Sensors* **2024**, *24*, 2626, doi:10.3390/s24082626.
66. Wang, C.; Finamore, A.; Michiardi, P.; Gallo, M.; Rossi, D. Data Augmentation for Traffic Classification. In *Passive and Active Measurement*; Richter, P., Bajpai, V., Carisimo, E., Eds.; Lecture Notes in Computer Science; Springer Nature Switzerland: Cham, 2024; Vol. 14537, pp. 159–186 ISBN 978-3-031-56248-8.
67. Song, Q.; Liu, J.; Zhang, C.; Wang, Z.; Wei, Z.; Ma, P. Classification of IPSec VPN Encrypted Traffic Based on a Hybrid Method. In Proceedings of the 2024 2nd International Conference on Electronics, Computers and Communication Technology; ACM: Chengdu, China, October 25, 2024; pp. 204–209.
68. Gudla, R.; Vollala, S.; Srinivasa, K.G.; Amin, R. A Novel Approach for Classification of Tor and Non-Tor Traffic Using Efficient Feature Selection Methods. *Expert Systems with Applications* **2024**, *249*, 123544, doi:10.1016/j.eswa.2024.123544.
69. Chong, K. Spatiotemporal Influence Analysis Through Traffic Speed Pattern Analysis Using Spatial Classification. *Applied Sciences* **2024**, *15*, 196, doi:10.3390/app15010196.
70. Krupski, J.; Iwanowski, M.; Graniszewski, W. On the Right Choice of Data from Popular Datasets for Internet Traffic Classification. *Computer Communications* **2025**, *233*, 108068, doi:10.1016/j.comcom.2025.108068.
71. Xie, J.; Li, S.; Yun, X.; Si, C.; Yin, T. Sample Analysis and Multi-Label Classification for Malicious Sample Datasets. *Computer Networks* **2025**, *258*, 110999, doi:10.1016/j.comnet.2024.110999.
72. Zhan, M.; Yang, J.; Jia, D.; Fu, G. EAPT: An Encrypted Traffic Classification Model via Adversarial Pre-Trained Transformers. *Computer Networks* **2025**, *257*, 110973, doi:10.1016/j.comnet.2024.110973.

73. Li, Z.; Bu, L.; Wang, Y.; Ma, Q.; Tan, L.; Bu, F. Hierarchical Perception for Encrypted Traffic Classification via Class Incremental Learning. *Computers & Security* **2025**, *149*, 104195, doi:10.1016/j.cose.2024.104195.
74. Xu, S.-J.; Kong, K.-C.; Jin, X.-B.; Geng, G.-G. Unveiling Traffic Paths: Explainable Path Signature Feature-Based Encrypted Traffic Classification. *Computers & Security* **2025**, *150*, 104283, doi:10.1016/j.cose.2024.104283.
75. Han, Y.; Huang, Z.; Xu, P. Field-Road Trajectory Classification for Agricultural Machinery by Integrating Spatio-Temporal Clustering and Semantic Segmentation. *Computers and Electronics in Agriculture* **2025**, *233*, 110139, doi:10.1016/j.compag.2025.110139.
76. Liu, Z.; Wei, Q.; Song, Q.; Duan, C. Fine-Grained Encrypted Traffic Classification Using Dual Embedding and Graph Neural Networks. *Electronics* **2025**, *14*, 778, doi:10.3390/electronics14040778.
77. Zhu, L.; Zhang, Q.; Jian, X.; Yang, Y. Graph Convolutional Network for Traffic Incidents Duration Classification. *Engineering Applications of Artificial Intelligence* **2025**, *151*, 110570, doi:10.1016/j.engappai.2025.110570.
78. Abbasi, M.; López Flórez, S.; Shahraki, A.; Taherkordi, A.; Prieto, J.; Corchado, J. M. Class Imbalance in Network Traffic Classification: An Adaptive Weight Ensemble-of-Ensemble Learning Method. *IEEE Access* **2025**, *13*, 26171–26192, doi:10.1109/ACCESS.2025.3538170.
79. Lee, S.-H.; Lee, S.; Yun, I. Mosaic-Mixed Attention-Based Unexpected Traffic Scene Classification. *IEEE Access* **2025**, *13*, 15712–15722, doi:10.1109/ACCESS.2025.3531121.
80. Mezina, A.; Nurmi, J.; Ometov, A. Novel Hybrid UNet++ and LSTM Model for Enhanced Attack Detection and Classification in IoMT Traffic. *IEEE Access* **2025**, 1–1, doi:10.1109/ACCESS.2025.3553966.
81. Ouyang, G.; Guo, Y.; Lu, Y.; He, F. Incremental Learning for Network Traffic Classification Using Generative Adversarial Networks. *IEICE Trans. Inf. & Syst.* **2025**, *E108.D*, 124–136, doi:10.1587/transinf.2024EDP7129.
82. Wang, Y.; Song, L. Application and Optimization of Convolutional Neural Networks Based on Deep Learning in Network Traffic Classification and Anomaly Detection. *IJCAI* **2025**, *49*, doi:10.31449/inf.v49i14.7602.
83. Guan, X.; Zhu, S.; Wan, X.; Wu, Y. STARNeT: Multidimensional Spatial–Temporal Attention Recall Network for Accurate Encrypted Traffic Classification. *Journal of Network and Computer Applications* **2025**, *236*, 104109, doi:10.1016/j.jnca.2025.104109.
84. Li, Z.; Zhao, H.; Zhao, J.; Jiang, Y.; Bu, F. SAT-Net: A Staggered Attention Network Using Graph Neural Networks for Encrypted Traffic Classification. *Journal of Network and Computer Applications* **2025**, *233*, 104069, doi:10.1016/j.jnca.2024.104069.
85. Wang, Z.; Lin, W.; Chen, Y.; Vai, M. I. Robust Classification of Encrypted Network Services Using Convolutional Neural Networks Optimized by Information Bottleneck Method. *IEEE Access* **2025**, *13*, 36995–37005, doi:10.1109/ACCESS.2025.3545476.
86. Xu, S.; Han, J.; Liu, Y.; Liu, H.; Bai, Y. Few-Shot Traffic Classification Based on Autoencoder and Deep Graph Convolutional Networks. *Sci. Rep.* **2025**, *15*, 8995, doi:10.1038/s41598-025-94240-6.
87. Zai-jian, W.; Dong, Y.; Shi, H.; Lingyun, Y.; Pingping, T. Internet Video Traffic Classification Using QoS Features. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC); IEEE: Kauai, HI, USA, February 2016; pp. 1–5.

88. Dong, Y.; Zhao, J.; Jin, J. Novel Feature Selection and Classification of Internet Video Traffic Based on a Hierarchical Scheme. *Computer Networks* **2017**, *119*, 102–111, doi:10.1016/j.comnet.2017.03.019.
89. Canovas, A.; Jimenez, J. M.; Romero, O.; Lloret, J. Multimedia Data Flow Traffic Classification Using Intelligent Models Based on Traffic Patterns. *IEEE Network* **2018**, *32*, 100–107, doi:10.1109/MNET.2018.1800121.
90. Hayashi, K.; Ooka, R.; Miyoshi, T.; Yamazaki, T. P2PTV Traffic Classification and Its Characteristic Analysis Using Machine Learning. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS); IEEE: Matsue, Japan, September 2019; pp. 1–6.
91. Wu, H.; Li, X.; Cheng, G.; Hu, X. Monitoring Video Resolution of Adaptive Encrypted Video Traffic Based on HTTP/2 Features. In Proceedings of the IEEE INFOCOM 2021 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); IEEE: Vancouver, BC, Canada, May 10, 2021; pp. 1–6.
92. Ozkan, H.; Temelli, R.; Gurbuz, O.; Koksal, O. K.; Ipekoren, A. K.; Canbal, F.; Karahan, B. D.; Kuran, M. Ş. Multimedia Traffic Classification with Mixture of Markov Components. *Ad Hoc Networks* **2021**, *121*, 102608, doi:10.1016/j.adhoc.2021.102608.
93. Wu, Z.; Dong, Y.; Jin, J.; Wei, H.-L.; Xie, G. Multimedia Traffic Classification for Imbalanced Environment. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 1838–1852, doi:10.1109/TNSE.2022.3153925.
94. Bukhari, S. M. A. H.; Afaq, M.; Song, W.-C. PPS: A Packets Pattern-Based Video Identification in Encrypted Network Traffic. In Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing; ACM: Taormina (Messina), Italy, December 4, 2023; pp. 1–6.
95. Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Information Fusion* **2020**, *58*, 82–115, doi:10.1016/j.inffus.2019.12.012.
96. Khani, P.; Moeinaddini, E.; Abnavi, N. D.; Shahraki, A. Explainable Artificial Intelligence for Feature Selection in Network Traffic Classification: A Comparative Study. *Trans. Emerging Tel. Tech.* **2024**, *35*, e4970, doi:10.1002/ett.4970.
97. Dong, H.; Lee, J. S. A. The Metaverse From a Multimedia Communications Perspective. *IEEE MultiMedia* **2022**, *29*, 123–127, doi:10.1109/MMUL.2022.3217627.
98. Carrascosa, M.; Bellalta, B. Cloud-Gaming: Analysis of Google Stadia Traffic. *Computer Communications* **2022**, *188*, 99–116, doi:10.1016/j.comcom.2022.03.006.
99. Wang, Y.; Su, Z.; Zhang, N.; Xing, R.; Liu, D.; Luan, T. H.; Shen, X. A Survey on Metaverse: Fundamentals, Security, and Privacy. *IEEE Commun. Surv. Tutorials* **2023**, *25*, 319–352, doi:10.1109/COMST.2022.3202047.
100. Martin-Perez, J.; Cominardi, L.; Bernardos, C. J.; De La Oliva, A.; Azcorra, A. Modeling Mobile Edge Computing Deployments for Low Latency Multimedia Services. *IEEE Trans. on Broadcast.* **2019**, *65*, 464–474, doi:10.1109/TBC.2019.2901406.
101. Chen, M.; Saad, W.; Yin, C.; Debbah, M. Data Correlation-Aware Resource Management in Wireless Virtual Reality (VR): An Echo State Transfer Learning Approach. *IEEE Trans. Commun.* **2019**, *67*, 4267–4280, doi:10.1109/TCOMM.2019.2900624.

102. Perfecto, C.; Elbamby, M. S.; Ser, J. D.; Bennis, M. Taming the Latency in Multi-User VR 360°: A QoE-Aware Deep Learning-Aided Multicast Framework. *IEEE Trans. Commun.* **2020**, *68*, 2491–2508, doi:10.1109/TCOMM.2020.2965527.
103. Lecci, M.; Drago, M.; Zanella, A.; Zorzi, M. An Open Framework for Analyzing and Modeling XR Network Traffic. *IEEE Access* **2021**, *9*, 129782–129795, doi:10.1109/ACCESS.2021.3113162.
104. Alencar, D.; Both, C.; Antunes, R.; Oliveira, H.; Cerqueira, E.; Rosario, D. Dynamic Microservice Allocation for Virtual Reality Distribution With QoE Support. *IEEE Trans. Netw. Serv. Manage.* **2022**, *19*, 729–740, doi:10.1109/TNSM.2021.3076922.
105. Chen, T.; Grabs, E.; Tasic, I.; Cano, M.-D. Network Traffic Analysis for eXtended Reality Applications. In Proceedings of the XVI Jornadas de Ingenieria Telematica (JITEL 2023); 2023.
106. Chiariotti, F.; Drago, M.; Testolina, P.; Lecci, M.; Zanella, A.; Zorzi, M. Temporal Characterization and Prediction of VR Traffic: A Network Slicing Use Case. *IEEE Trans. on Mobile Comput.* **2023**, 1–18, doi:10.1109/TMC.2023.3282689.
107. Balasubramanian, V.; Bouachir, O.; Al-Habashnat, A. BOLT: A Bayesian Online Learning Framework for Time Sensitive Networks in Metaverse. In Proceedings of the 2023 International Conference on Intelligent Metaverse Technologies & Applications (iMETA); IEEE: Tartu, Estonia, September 18, 2023; pp. 1–7.
108. Manjunath, Y. S. K.; Szymanowski, M.; Wissborn, A.; Li, M.; Zhao, L.; Zhang, X.-P. ResLearn: Transformer-Based Residual Learning for Metaverse Network Traffic Prediction 2024.
109. Sani, Y.; Mauthe, A.; Edwards, C. Adaptive Bitrate Selection: A Survey. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2985–3014, doi:10.1109/COMST.2017.2725241.
110. Van Der Auwera, G.; David, P.; Reisslein, M. Traffic Characteristics of H.264/AVC Variable Bit Rate Video. *IEEE Commun. Mag.* **2008**, *46*, 164–174, doi:10.1109/MCOM.2008.4689260.
111. *Livestreaming Setup and Tools*; Available online: https://help.vimeo.com/hc/en-us/sections/12397317675665-Livestreaming-setup-and-tools
112. *Live Stream on YouTube*; Available online: https://support.google.com/youtube/topic/9257891
113. *Broadcasting Guidelines*; Available online: https://help.twitch.tv/s/article/broadcasting-guidelines
114. *Media Transfer Guide*; Available online: https://developers.tiktok.com/doc/content-posting-api-media-transfer-guide
115. *Requirements for Facebook Live Videos*; Available online: https://www.facebook.com/business/help/162540111070395
116. *Content Delivery Networks*; Buyya, R., Pathan, M., Vakali, A., Eds.; Lecture Notes Electrical Engineering; Springer Berlin Heidelberg: Berlin, Heidelberg, 2008; Vol. 9; ISBN 978-3-540-77886-8.
117. Liatifis, A. L.; Tegos, S. A. T.; Mitsiou, N. A. M.; Makris, I. M.; Kyranou, K. K.; Pliatsios, D. P.; Lagkas, T. L.; Sarigiannidis, P. S. NANCY SNS JU Project “Greek In-Lab Testbed Dataset 1.”
118. Beccari, M. B.; Clavenna, A. C.; Albanese, A. A. NANCY SNS-JU Project “Italian in-Lab Testbed Dataset 1” D6.6.
119. Beccari, M. B.; Clavenna, A. C.; Albanese, A. A. NANCY SNS-JU Project “Italtel Italian in-Lab Testbed Dataset 2.”
120. Pantos, R.; May, W. *HTTP Live Streaming*; RFC Editor, 2017; p. RFC8216;.

121. Reznik, Y.; Cabrera, G. Multi-Regional, Multi-CDN Delivery Optimizations Using HLS/DASH Content Steering Standard. In Proceedings of the Proceedings of the 4th Mile-High Video Conference; ACM: Denver, CO, USA, February 18, 2025; pp. 7–12.
122. *Simple Realtime Server (SRS)*; Available online: https://github.com/ossrs/srs
123. *Big Buck Bunny*; Available online: https://peach.blender.org
124. Tomar, S. Converting Video Formats with FFmpeg. *Linux J.* **2006**, *2006*, 10.
125. Müller, C.; Timmerer, C. A VLC Media Player Plugin Enabling Dynamic Adaptive Streaming over HTTP. In Proceedings of the 19th ACM International Conference on Multimedia; ACM: Scottsdale, Arizona, USA, November 28, 2011; pp. 723–726.
126. O'Hanlon, P.; Aslam, A. Latency Target-Based Analysis of the DASH.Js Player. In Proceedings of the 14th ACM Multimedia Systems Conference; ACM: Vancouver, BC, Canada, June 7, 2023; pp. 153–160.
127. Mahmoud, H.; Abozariba, R. A Systematic Review on WebRTC for Potential Applications and Challenges beyond Audio Video Streaming. *Multimed. Tools Appl.* **2024**, *84*, 2909–2946, doi:10.1007/s11042-024-20448-9.
128. Chakraborty, T.; Chattopadhyay, S.; Das, S.; Kumar, U.; Senthilnath, J. Searching for Heavy-Tailed Probability Distributions for Modeling Real-World Complex Networks. *IEEE Access* **2022**, *10*, 115092–115107, doi:10.1109/ACCESS.2022.3218631.
129. Grab, E.; Petersons, E. Hurst Parameter Estimation by Wavelet Transformation and a Filter Bank for Self-Similar Traffic. *Aut. Control Comp. Sci.* **2015**, *49*, 286–292, doi:10.3103/S0146411615050041.
130. Franzl, G. Queueing Models for Multi-Service Networks. PhD Thesis, Technische Universität Wien, 2015.
131. Lin, K.; Xu, X.; Xiao, F. MFFusion: A Multi-Level Features Fusion Model for Malicious Traffic Detection Based on Deep Learning. *Computer Networks* **2022**, *202*, 108658, doi:10.1016/j.comnet.2021.108658.
132. Song, M.; Ran, J.; Li, S. Encrypted Traffic Classification Based on Text Convolution Neural Networks. In Proceedings of the 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT); IEEE: Dalian, China, October 2019; pp. 432–436.
133. Li, F.; Ye, F. Simplifying Data Traffic Classification with Byte Importance Distillation. In Proceedings of the ICC 2021 – IEEE International Conference on Communications; IEEE: Montreal, QC, Canada, June 2021; pp. 1–6.
134. Roy, S.; Shapira, T.; Shavitt, Y. Fast and Lean Encrypted Internet Traffic Classification. *Computer Communications* **2022**, *186*, 166–173, doi:10.1016/j.comcom.2022.02.003.
135. Yang, Y.; Yan, Y.; Gao, Z.; Rui, L.; Lyu, R.; Gao, B.; Yu, P. A Network Traffic Classification Method Based on Dual-Mode Feature Extraction and Hybrid Neural Networks. *IEEE Trans. Netw. Serv. Manage.* **2023**, *20*, 4073–4084, doi:10.1109/TNSM.2023.3262246.
136. Batista, G. E. A. P. A.; Prati, R. C.; Monard, M. C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.* **2004**, *6*, 20–29, doi:10.1145/1007730.1007735.
137. Lemaître, G.; Nogueira, F.; Aridas, C. K. Imbalanced-Learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* **2017**, *18*, 1–5.
138. Hofstede, R.; Celeda, P.; Trammell, B.; Drago, I.; Sadre, R.; Sperotto, A.; Pras, A. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 2037–2064, doi:10.1109/COMST.2014.2321898.

139. Brownlee, N.; Mills, C.; Ruth, G. *Traffic Flow Measurement: Architecture*; RFC Editor, 1999; p. RFC2722.
140. Yu, L.; Tao, J.; Xu, Y.; Sun, W.; Wang, Z. TLS Fingerprint for Encrypted Malicious Traffic Detection with Attributed Graph Kernel. *Computer Networks* **2024**, *247*, 110475, doi:10.1016/j.comnet.2024.110475.
141. Aouini, Z.; Pekar, A. NFStream: A Flexible Network Data Analysis Framework. *Computer Networks* **2022**, *204*, 108719, doi:10.1016/j.comnet.2021.108719.
142. Chen, T.; Grabs, E.; Ipatovs, A.; Cano, M.-D. A Novel Proprietary Internet Video Traffic Dataset Generation Algorithm. *Applied Sciences* **2025**, *15*, 515, doi:10.3390/app15020515.
143. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-End Encrypted Traffic Classification with One-Dimensional Convolution Neural Networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI); IEEE: Beijing, China, July 2017; pp. 43–48.
144. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API Design for Machine Learning Software: Experiences from the Scikit-Learn Project. **2013**, doi:10.48550/ARXIV.1309.0238.
145. Van Der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J. D.; Yager, N.; Gouillart, E.; Yu, T.; the scikit-image contributors Scikit-Image: Image Processing in Python 2014.
146. Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* **2000**.
147. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. **2016**, doi:10.48550/ARXIV.1603.02754.
148. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library 2019.
149. Fey, M.; Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric 2019.
150. Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R.; Saberian, M. Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. *Soft Comput.* **2020**, *24*, 1999–2012, doi:10.1007/s00500-019-04030-2.
151. Shapira, T.; Shavitt, Y. FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification. *IEEE Trans. Netw. Serv. Manage.* **2021**, *18*, 1218–1232, doi:10.1109/TNSM.2021.3071441.
152. Liu, C.; He, L.; Xiong, G.; Cao, Z.; Li, Z. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. In Proceedings of the IEEE INFOCOM 2019 – IEEE Conference on Computer Communications; IEEE: Paris, France, April 2019; pp. 1171–1179.
153. Martinsson, P.-G.; Rokhlin, V.; Tygert, M. A Randomized Algorithm for the Decomposition of Matrices. *Applied and Computational Harmonic Analysis* **2011**, *30*, 47–68, doi:10.1016/j.acha.2010.02.003.
154. Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization 2014.
155. Mao, A.; Mohri, M.; Zhong, Y. Cross-Entropy Loss Functions: Theoretical Analysis and Applications 2023.
156. Dorogush, A. V.; Ershov, V.; Gulin, A. CatBoost: Gradient Boosting with Categorical Features Support 2018.
157. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer Texts in Statistics; Springer New York: New York, NY, 2013; Vol. 103; ISBN 978-1-4614-7137-0.

ACKNOWLEDGMENT

First, I would like to express my sincere gratitude to my scientific supervisors, Associate Professor Elans Grabs and Associate Professor Aleksandrs Ipatovs. I have worked with Associate Professor Elans Grabs since my master's studies, and I am deeply thankful for his meticulous guidance, which not only strengthened my understanding of the subject but also inspired me to expand my research into this doctoral thesis. His continuous support and advice throughout my doctoral studies have been invaluable, both academically and personally.

I am also grateful to Associate Professor Aleksandrs Ipatovs for his assistance in securing projects and funding, which allowed me to integrate my doctoral research with scientific initiatives. This greatly enriched the depth of the study and enhanced its practical value.

My sincere thanks to Professor Cano Baños María Dolores, who kindly agreed to serve as my co-advisor. Her guidance helped me overcome research challenges, align my work with current research trends, and adapt my contributions to the evolving needs of the telecommunications industry. I also appreciate the European University of Technology (EUt+) alliance for fostering collaboration and enabling valuable research connections.

I would like to thank Professor Vjačeslavs Bobrovs for providing the opportunity to pursue doctoral studies in telecommunications at Riga Technical University and for supporting my work as a researcher at the Institute of Photonics, Electronics and Telecommunications. I am equally grateful to the Institute, as well as to all colleagues and teachers whose advice and encouragement have greatly contributed to my academic growth. Special thanks are extended to Anna Sedova, Nataļja Muračova, and Mārīte Traniņa for their kind assistance during my doctoral studies.

Finally, I owe heartfelt gratitude to my parents and grandparents for their unwavering material and emotional support. I am also thankful to the people I met in Latvia, whose friendship enriched my journey, and to Latvia itself for becoming an unforgettable part of my life.

SUPPLEMENTS

[Publication 1]: Ensemble Learning Enabled Flow-level Internet Traffic Classification

Chen, T., Grabs, E., Ipatovs, A., Cano M. “Ensemble Learning Enabled Flow-level Internet Traffic Classification” submitted to Journal of Information and Telecommunication (Under review).

[Publication 2]: Video Traffic Classification in the 5G Open RAN Network

Chen, T., Benkis R., Bogdanovs, N., Stetjuha M., Klūga J., Jeralovičs V., Rjazanovs D., Grabs, E., Ipatovs, A. “Video Traffic Classification in the 5G Open RAN Network” submitted to 2025 Photonics & Electromagnetics Research Symposium (PIERS 2025) Conference (Accepted), United Arab Emirates, Abu Dhabi, 4-8 May, 2025.

[Publication 3]: A Novel Proprietary Internet Video Traffic Dataset Generation Algorithm

Chen, T., Grabs, E., Ipatovs, A., Cano, M. A Novel Proprietary Internet Video Traffic Dataset Generation Algorithm. Applied Sciences, Vol. 15, No. 2, Article number 515. e-ISSN 2076-3417, 2025.

applied sciences

MDPI

Article

A Novel Proprietary Internet Video Traffic Dataset Generation Algorithm

Tianhua Chen [1,*], Elans Grabs [1], Aleksandrs Ipatovs [1] and Maria-Dolores Cano [2,*]

[1] Institute of Photonics, Electronics and Telecommunications, Riga Technical University, LV-1048 Riga, Latvia; elans.grabs@rtu.lv (E.G.); aleksandrs.ipatovs@rtu.lv (A.I.)
[2] Department of Information Technologies and Communication, Universidad Politécnica de Cartagena, Plaza del Hospital 1, 30202 Cartagena, Spain
* Correspondence: tianhua.chen@rtu.lv (T.C.); mdolores.cano@upct.es (M.-D.C.)

Abstract: Considering the exponential growth of network traffic, particularly driven by over-the-top (OTT) streaming applications, video category network traffic constitutes a significant portion of overall network traffic. However, most research has focused on the categorization and diversity of network traffic using benchmark datasets, with limited attention paid to video category network traffic. Additionally, there is a lack of proprietary Internet video traffic datasets, and the few proprietary datasets available often lack transparency and interpretability. This paper introduces a novel framework for generating proprietary Internet video traffic datasets, addressing existing gaps in dataset quality and consistency. We propose the nYFTQC algorithm, which enables the creation of fifteen detailed datasets specifically designed for Internet video traffic analysis. The proposed datasets demonstrate superior performance metrics, including completeness, consistency, and transparency. This comprehensive approach enhances the accuracy and interpretability of traffic sample analysis, providing valuable resources for future research in video category network traffic.

Keywords: network traffic classification; proprietary dataset; algorithm; interpretability

Academic Editor: Andrea Prati

Received: 3 December 2024
Revised: 3 January 2025
Accepted: 5 January 2025
Published: 7 January 2025

Citation: Chen, T.; Grabs, E.; Ipatovs, A.; Cano, M.-D. A Novel Proprietary Internet Video Traffic Dataset Generation Algorithm. *Appl. Sci.* **2025**, *15*, 515. https://doi.org/10.3390/app15020515

1. Introduction

The volume of network traffic is increasing exponentially, largely driven by the growing use of over-the-top (OTT) media services [1]. These video applications generate significant network traffic characterized by large packet volumes, vast data bytes, and irregular packet arrival intervals. As a result, video traffic has become a critical focus of analysis in Network Traffic Monitoring and Classification (NTMC). The benchmark Internet traffic flow dataset BRASIL: Characterizing Network-based Applications was introduced in 2005 and classified Windows Media Player and Real applications under the multimedia category [2]. Similarly, popular benchmark datasets such as ISCXVPN2016, ISCXTor2016, and CIC-Darknet2020 categorize video traffic under labels like Streaming and Video-Stream [3–5]. However, categories such as Web Browsing, Chat, Transfer, Audio-Stream, and VoIP may include traffic flows related to video, but the boundaries between applications and categories remain unclear, requiring further validation for accurate classification. In modern networks, video traffic resists straightforward categorization, it is intertwined with other traffic categories in diverse characteristics, including network protocols, architectures, application types, and traffic distribution characteristics, creating a complex profile. While some datasets include video-related traffic, these samples typically represent only a small portion of the total dataset. To address this, researchers have developed proprietary datasets

tailored for specific traffic categories. This paper reviews existing network traffic datasets that contain video traffic and introduces a novel algorithm to generate a proprietary Internet video traffic dataset. The proposed approach aims to improve the performance of classification metrics in the video traffic category within NTMC.

2. Related Works

In this section, we will comprehensively compare the characteristics of video-type traffic across various public and proprietary network traffic datasets, including category and application selection, transmission protocol, sample size ratio, etc. The BRASIL dataset contains a total of 14 categories, with over one million traffic flow records; however, multimedia traffic accounts for less than 0.1% [2]. The ISCXVPN2016 and ISCXTor2016 datasets categorize traffic into 14 groups based on VPN and Tor characteristics, respectively. Each dataset includes seven main categories, with the video category featuring streaming applications such as Vimeo, Facebook, and YouTube. Both ISCXVPN2016 and ISCXTor2016 contain over 150,000 flow samples, with video-streaming traffic accounting for less than 5% and 2%, respectively [3,4]. The CIC-Darknet2020 dataset merges the VPN and Tor traffic from the ISCXVPN2016 and ISCXTor2016 datasets into a Darknet category, with video-streaming traffic still accounting for less than 5% of over 150,000 flow samples [5]. The traffic categories in these datasets are often selected based on specific tasks in the NTMC, such as intrusion detection, botnet detection, encrypted traffic analysis, and anonymized network traffic studies.

Bader et al. developed the OSF-EIMTC traffic classification framework, which demonstrated excellent performance and was validated on several public and benchmark datasets, including USTC-TFC2016, ISCXVPN2016, Ariel (BOA2016), MAppGraph2021, MTA (Malware Traffic Analysis), and StratosphereIPS [6]. The USTC-TFC2016 dataset includes 10 categories of malware traffic and 10 categories of benign traffic, with the video-related category containing applications such as FaceTime and Skype. This dataset includes over 75,000 flow and session samples [7]. The StratosphereIPS dataset includes only three categories, benign, malware, and mixed, without application or protocol metadata. The MTA dataset contains two categories, malware and legitimate traffic, both lacking application and protocol metadata. The Ariel dataset contains over 20,000 session samples across 30 categories, with video traffic from YouTube and Facebook accounting for less than 10% of the total dataset [8]. The MAppGraph dataset provides over one million flow records spanning 101 mobile application categories. Its video traffic category includes applications such as Netflix, TikTok, and Twitch, which contribute less than 10% of the total dataset [9]. Liu et al. proposed the Multi-Task Learning Fusion (MTEFU) algorithm, achieving 94.67% classification accuracy on the QUIC dataset, which includes five service categories: Google Docs, Google Drive, Google Music, YouTube, and Google Search. The dataset comprises nearly 7000 flow records [10]. Existing studies show that video traffic samples make up only a small portion of most public and benchmark datasets. Key issues include limited sample sizes, narrowly defined categories, and a lack of metadata descriptions. Additionally, these datasets often employ inconsistent naming conventions for applications, categories, SSL server names, Server Name Indication (SNI), and other elements, complicating the validation of relationships between samples and their naming conventions.

Some proprietary Internet traffic datasets focus on the video traffic category. Salman et al. utilized packet-level and flow-level features such as size, inter-arrival time, and packet flow direction, combined with ensemble deep learning classifiers, achieving 89.77% classification accuracy across four traffic categories: Interactive, Bulk Data Transfer, Streaming, and Transaction. Their dataset included 3612 flow records [11]. Arestrom et al. categorized video traffic into six groups based on Quality of Service (QoS) and Quality of Experience

(QoE) characteristics: video streaming, web browsing, social networking, audio communication, text communication, and bulk download. Their dataset included 7154 session samples [12]. Wu et al. leveraged byte and time sequence features to classify traffic from 20 Google and Apple applications and 58 web services, achieving over 98% accuracy using the TLS Flow Sequence Network (TFSN) model on a dataset of 13,915 TLS flow samples [13]. Xiao et al. proposed the Extended Byte Segment Neural Network (EBSNN) with long short-term memory (LSTM) for traffic classification across 20 websites, achieving 99.96% accuracy. They collected two kinds of datasets, application identification and website identification with 29 and 20 categories, respectively, and also chose a total of 11,607 flow records from 8 application categories for validation [14]. Gabilondo et al. defined five application traffic categories, Best-Effort/Default, Control Data, Video, IoT, and WebData, analyzing bandwidth requirements and QoS performance across 5G network slice scenarios based on service types and QoS-related characteristics [15].

Wang et al. categorized video traffic into five groups based on QoS levels: broadcast video (BV), web video (WV), trade-style video (TSV), barter-style video (BSV), and interactive video (IV). Each category contained 100 flow samples, sourced from platforms such as Tudou, Youku, TVAnt, Skype, and others. Using a modified K-Singular Value Decomposition (KSVD) classification framework, they achieved 98.97% classification accuracy [16]. Dong et al. employed a hierarchical k-Nearest Neighbor (k-NN) algorithm and achieved over 96.77% classification accuracy on their Internet video traffic dataset. This dataset included six categories: asymmetric standard-definition videos (ASD), asymmetric high-definition videos (AHD), HTTP-download videos (HDV), interactive videos (CV), P2P_video, and network live TV (ILV). It comprised 360 flow samples with a total data volume of 13.03 GB [17]. Canovas et al. developed a multimedia traffic classification system based on objective QoE, collecting 2741 video traffic flow samples labeled into five categories: Non-critical, Low-critical, Some-critical, Critical, and Very-critical [18]. Hayashi et al. collected 400 peer-to-peer video streaming (P2PTV) traffic samples and classified them into server-based bursty traffic, stable traffic, and peer-based bursty traffic [19]. Arestrom et al. achieved an F1-score of 0.893 in classifying video traffic across two categories, Video-on-Demand (VoD) and live streaming, using a dataset of 1232 flow samples [12]. Costa Da Silva et al. defined the video category as LowVideo (Lv), AverageVideo (Av), and HighVideo (Hv) based on the chance of detection of a potential characterization trained in flows fuzzy classifiers achieved as high as 90% classification accuracy; however, the datasets used in their study were not adequately described [20]. The above works have developed proprietary Internet video traffic datasets and provided in-depth analyses of video traffic. However, several issues persist. Video traffic often originates from different web platforms and applications or relies on privately emulated video traffic. This makes the classification of Internet video traffic into categories more subjective, as it depends on varying video traffic behaviors or attribute information.

Wu et al. enhanced Internet video traffic classification by introducing the SV category, further subdivided into Standard Definition (480P, SD), High Definition (720P, HD), and Ultra-clear Definition (1080P, UD), achieving 95% accuracy across seven categories using a Chain Hierarchical Structure (CHS) scheme. The dataset included 25,200 flow records [21]. In another study, Wu et al. classified streaming video traffic based on raw video resolution, including audio, 144P, 240P, 360P, 480P, and 720P, into six categories. Using transport layer-encapsulated chunk data unit sizes as features with a Convolutional neural network (CNN) model, they achieved 99% accuracy for each category and a minimum F1-score of 95% on a dataset of 11,510 flow samples [22]. Bukhari et al. developed a proprietary YouTube video dataset containing 5000 streams from 100 categories across different channels, achieving 90% classification accuracy by combining the CNN model with

Packets-Per-Second (PPS) features [23]. Ozkan et al. evaluated video traffic classification across seven popular mobile applications: Netflix, YouTube, YouTube Live, Twitch, Spotify, WhatsApp, and Skype, using statistical models such as Mixture of Markov Components (MMC), k-Nearest Markov Component (kNMC), and k-Nearest Markov Parameter (kNMP). Their dataset, approximately 6GB in size, covered four categories: Video on Demand (VoD), Video Live Streaming (VLS), Sharing and Media Streaming (S&MS), and Teleconferencing (TC) [24].

In general, whether using public or proprietary Internet traffic datasets, video traffic is typically treated as a distinct category, though the labeling definitions for this category vary significantly. Researchers consider several factors when defining labels. First, they consider a priori knowledge about the video, including the web platform, application, content labels, source encoding, resolution, and refresh rate. Second is metadata from the video transmission process, such as transport protocols (e.g., Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Quick UDP Internet Connections (QUIC)) and streaming technologies (e.g., Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), VoD, and VLS). Additional factors include QoS and QoE metrics, as well as the utilization of Tor, VPNs, and other technologies. These elements impose higher demands for achieving fine-grained network video traffic classification. However, most existing proprietary Internet video traffic datasets still have some limitations, including non-disclosure of the dataset, small sample sizes, weak correlations between samples and the aforementioned video-related factors, and limited interpretability. To address these gaps, we develop three algorithms for generating proprietary Internet video traffic datasets.

The main contributions of this paper can be summarized as follows:

- A novel and explainable proprietary Internet video traffic dataset generation algorithm is proposed, which generates fifteen corresponding datasets.
- The resulting datasets demonstrate excellent performance metrics, including completeness, consistency, and transparency.

The rest of this paper is organized as follows: Section 3 outlines the research methodology, covering the NFStream preprocessing algorithm, the proprietary Internet video traffic dataset generation algorithm, and the CICFlowmeter preprocessing algorithm. Section 4 provides an analysis of the proprietary Internet video traffic datasets, focusing on imbalance ratios, video traffic histogram density, and feature correlation. Section 5 concludes the paper.

3. Methodology

Our proposed proprietary Internet video traffic dataset generation framework is illustrated in Figure 1. First, this approach involves streaming a single web video on a specific platform for 30 min using Google Chrome (version 100.0.4896.88). During this process, wireless traffic from an Intel(R) Wi-Fi 6 AX200 160MHz network adapter is captured using Wireshark (version 3.4.3) under a Windows 10 (Build 18363.1316) operating system. The resulting Packet Capture (PCAP) files are labeled based on characteristics such as video resolution, frame rate, and VoD, or VLS attributes to facilitate distinction.

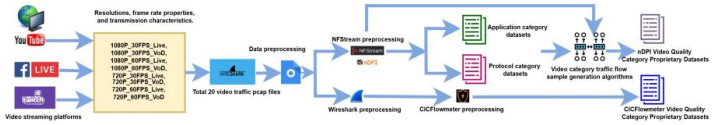


Figure 1. The framework of proprietary Internet video traffic dataset generation.

The next step involves data preprocessing, which includes NFStream preprocessing and Wireshark preprocessing. Sniffed traffic from a wireless adapter interface may include a mixture of traffic from various software or applications, and different analytical tools and methods can further refine the categorization of the dataset. The capabilities of network traffic analytic tools incorporate sniffing and capturing traffic packets and detecting network protocols, which are diffusely applied in cybersecurity monitoring and analysis. Prevalent network protocol and packet capture analysis software include Wireshark and Tcpdump, extensively deployed in network devices and systems at diverse system levels. Another category of traffic analysis tools has the main property of traffic classification. The representative approach is to perform traffic classification based on the results of Deep Packet Inspection (DPI) and generally utilized software tools, including L7-filter, OpenDPI, Libprotoident, nDPI, Zeek, Tstat, and NFStream [25,26]. Highly integrated traffic analytical tools can effectively accelerate the generation of proprietary datasets.

DPI techniques achieve traffic classification by parsing header fields in the application layer of Internet traffic, examining the entire transport layer payload, and extracting keywords from packet signatures or feature strings to identify web applications and protocol types. Typically utilized search and match algorithms include Aho–Corasick (AC) and Knuth–Morris–Pratt (KMP). With the popularity of encrypted traffic, DPI cannot extract the application layer payload information of encrypted traffic. Common solutions to this challenge include the lightweight inspection of limited packet payloads and metadata extraction. Metadata covers information such as bidirectional IP addresses and protocols in TCP/IP packets, and it also includes flow-related information, which, combined with powerful machine learning techniques, tremendously extends the direction of traffic classification. DPI technology is gradually evolving into Deep Flow Inspection (DFI) technology.

CICFlowmeter is a representative DFI tool capable of generating over 76 metadata features, which could be used as a reference for performance comparison [4]. The nDPI tool not only provides detected protocol references but also covers a large number of metadata discriminator conditions to furnish a guarantee for classification results [27]. In this paper, the actual utilization of NFStream [28], a traffic analysis tool based on nDPI and developed in Python language, has the advantage of being convenient to operate and extracting over 86 metadata as training features. We set the flow sample active timeout to 180 s for both tools. This configuration divides a single video traffic-related flow record into multiple records, simplifying further processing operations and increasing the sample volume. Additionally, we compare the differences in flow sample distributions under the same timeout value between the two tools.

In the final step, we integrate the NFStream preprocessing results with video category traffic flow generation algorithms to produce corresponding proprietary Internet video traffic datasets. Details are discussed in the following three subsections.

3.1. NFStream Preprocessing

In the NFStream preprocessing step, the process begins by sending pre-named PCAP files to the NFStream tool for traffic flow metering and exporting to CSV files. Based on the NFStream configuration's accounting mode, statistics are provided across four layers, IP, transport, link, and payload, resulting in over 800,000 flow records. In the second step, the NFStream tool, integrated with the nDPI library, detects the application name and application category for each flow, subsequently splitting them into two separate datasets based on these properties. These datasets require additional noise reduction and cleaning.

Figure 2 illustrates the flow record counts and distributions in the protocol and application category datasets across the four accounting modes. In the application category dataset, the Web category accounts for over 71% of the sample volume, followed by Net-

work, Download, and System. Similarly, in the protocol category dataset, the TLS category dominates with over 72% of the samples, while BitTorrent, DNS, and SSDP constitute the second-largest group. However, only 64% of the parsed flow records are categorized, leaving 36% of samples labeled as Unknown or Unspecified in the Protocol and Application categories, respectively. This occurs due to the 180-s active timeout setting, which truncates metering and may prevent some flows from accumulating enough metadata for accurate classification. Notably, the sample counts and distributions remain consistent across all accounting modes. The nDPI-parsed application name feature encompasses major and minor protocols, with minor protocols typically referring to specific software application names, such as Amazon, Google, Microsoft, etc. This study focuses on major protocols and excludes specific application software or serviced traffic flow samples.

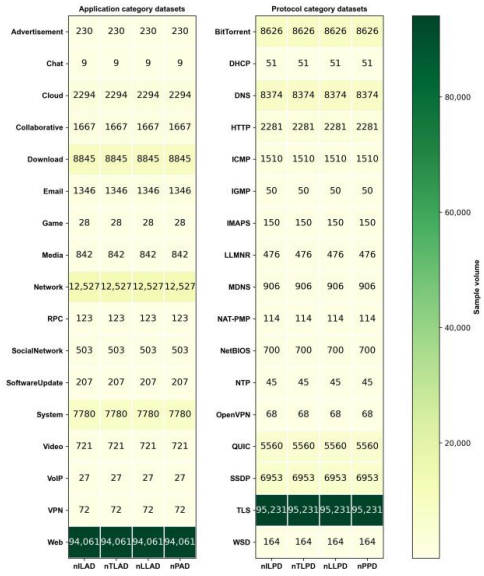


Figure 2. Protocol and application category datasets sample distribution.

Algorithm 1 outlines the nDPI-Detected Application Category and Protocol Category (nDACPC) dataset generation algorithm. A key step in this algorithm involves removing all flow samples where the detected application category is unspecified, as these flows cannot be assigned to a class in the given scenario. The next step counts the flow samples for each class and eliminates any class with fewer than six samples, as such classes cannot undergo further resampling operations.

Although both application category datasets and protocol category datasets detail the distribution of each class and the number of parsed flow samples, they are closely related. For instance, an application named TLS.Microsoft is generally classified under the Cloud category, while TLS.Microsoft365 is placed in the Collaborative category. Since Microsoft365 is a suite of Microsoft Office applications, it is understandable, upon human inspection,

why it falls under the Collaborative class. However, when the major protocol is the same, the subtle differences between minor protocols can make class determination challenging.

Algorithm 1 Pseudo-code for nDPI-Detected Application Category datasets and Protocol Category dataset generation (nDACPC)

```
Input: Each flow sample from segmented CSV files
Output: nILAD, nTLAD, nPAD, nLLAD, nILPD, nTLPD, nLLPD, nPPD dataset
for each CSV file in segmented CSV files do
    for each flow sample in CSV file do
        if Flow sample application name = "Unknown" then
            Delete the flow sample
        end if
        if Flow sample application category = "Unspecified" then
            Delete the flow sample
        end if
    end for
    Save all remaining samples
    for each flow sample in remaining samples do
        Read each sample application name major protocol
        Save read results into a new column 'protocol category'
    end for
    Count flow samples N for each protocol category class
    if N ≥ 6 then
        Keep all flow samples for the specific class
    else
        Delete all flow samples for the specific class
    end if
    Keep all remaining flow samples in the CSV file
end for
Merge different CSV files' samples into each dataset
Output nILAD, nTLAD, nPAD, nLLAD, nILPD, nTLPD, nLLPD, nPPD dataset
```

For example, in the case of DNS.Microsoft, application categories are divided into four main classes, Web, Cloud, System, and ConnCheck, based primarily on the request server names decoded by the nDPI library. One instance is the domain name www.msftconnecttest.com, used by Microsoft systems to check network connectivity, while the domain for Microsoft diagnostic data management services is events.data.microsoft.com. Similarly, the Microsoft account login domain includes login.live.com, and the Microsoft search engine domain is www.bing.com. Each application category can contain multiple domain names due to the diverse nature of the decoded domain names. Despite this variability, the application category can often be determined based on keywords within the domain name. If no domain names are decoded in the flow, the application category is determined by the minor protocol or a single protocol. The major protocol, minor protocol, and application category serve as ground-truth features used as labels for supervised training and interact with and reinforce each other. From this information, it is apparent that some service application traffic samples are mixed with video traffic samples, and there are still low-quality, irrelevant samples in the dataset. By integrating the characteristics of streamed videos and other prior knowledge, the existing proprietary datasets can be further improved and refined.

3.2. Proprietary Internet Video Traffic Dataset Generation

To generate proprietary video quality category datasets for YouTube, Facebook, and Twitch, we incorporated prior knowledge from PCAP files, including attributes such as video streaming platforms, resolution, and video playback modes. This knowledge was used to develop the

YouTube, Twitch, and Facebook video quality category datasets. Each dataset was further divided into four subsets based on four different nDPI accounting modes. Each dataset contains eight categories: 720P_30FPS_VoD, 720P_30FPS_Live, 720P_60FPS_VoD, 720P_60FPS_Live, 1080P_30FPS_Live, 1080P_30FPS_VoD, 1080P_60FPS_VoD, and 1080P_60FPS_Live. Due to limitations in capturing 60 frames per second on the Facebook platform, no packets for relevant 60 FPS videos were collected, resulting in only four 30 FPS classes being included for this platform.

The generation of nDPI-detected YouTube, Facebook, and Twitch video quality category datasets (nYFTQC) is outlined in Algorithm 2. The following are detailed steps.

Initial Filtering: The first step involves retaining all flow samples associated with the transport layer QUIC and TLS protocols, which primarily handle video traffic packets. For instance, the Google application YouTube interacts with Google's Chrome browser and servers using the QUIC protocol. Additionally, network video traffic is typically encrypted, with the TLS protocol widely used by users and video providers to ensure secure and reliable web communication.

Handling Unknown Categories: In the second step, flow samples from unknown application categories that the nDPI library cannot parse should be retained if they display certain characteristics, such as a high volume of bidirectional packets, large data sizes, and long durations of bidirectional flows. This retention specifically includes flow samples related to Google and Facebook.

Removing Irrelevant Flow Samples: Flow samples irrelevant to video content are deleted based on the requested server name. These include samples related to automatic updates, background processes, advertising statistics, account interfaces, and other non-video-related activities. Filtering is performed using the TLS application name and Web application category name.

Filtering Specific Protocols: The fourth step involves filtering out minor protocols associated with non-video traffic, such as those related to Microsoft, Skype, and similar applications, as indicated in the application name column. Using the elimination method, we delete flow records if the application category name is not from the following list: AmazonAWS, Amazon-Video, Azure, Twitch, Facebook, and YouTube. Additionally, flow samples are removed if the application category does not belong to one of these categories: Cloud, Media, Social Network, Video, Unspecified, and Web.

Excluding Inconsistent Flows: The fifth step removes flow samples with bidirectional flow durations of zero milliseconds or zero-count accumulators of bidirectional packets, as these are inconsistent with video traffic characteristics.

Final Compilation: The final step involves saving all remaining flow samples under a new column labeled Video Quality Category. These steps are repeated for all classes, and the remaining flow samples are merged to form a new dataset.

Overall, the process of selecting video-related flow samples using the nDPI tool is complex and meticulous. However, noise flow records that lack sufficient metadata for nDPI inspection or exhibit flow-level time–temporal features similar to normal samples are difficult to clean. Additional potential methods, such as flow aggregation or increasing the flow active timeout threshold, may help address this issue.

Algorithm 2 Pseudo-code for nDPI-detected YouTube, Facebook, and Twitch video Quality Category dataset generation (nYFTQC)

Input: Each flow sample from segmented CSV files
Output: YVAIL, YVATL, YVALL, YVAP, FVAIL, FVATL, FVALL, FVAD, TVAIL, TVATL, TVALL, TVAP dataset
for each CSV file in segmented CSV files **do**
    for each flow sample in CSV file **do**
        Read each PCAP file's prior knowledge
        Update each flow sample label
    end for
    Keep all remaining flow samples in the CSV file
end for
Update each CSV file label according to different platforms
Merger each segment CSV file into three platform categories CSV files
for Each CSV file in platform categorized CSV files **do**
    for Each flow sample in CSV file **do**
        if Flow sample application name major protocol $\neq$ "TLS", "QUIC", or "Unknown" **then**
            Delete the flow sample
        end if
        if Flow sample requested server name $\neq$ "Blanks", application name = "TLS" and application category name = "Web" **then**
            Delete the flow sample
        end if
        if Flow sample bidirectional duration or bidirectional packets = 0 **then**
            Delete the flow sample
        end if
    end for
    Save all remaining flow samples
    for Each flow sample in remaining samples **do**
        if Flow sample application category name $\neq$ "Cloud", "Media"; "SocialNetwork", "Video", "Unspecified", or "Web " **then**
            Delete the flow sample
        end if
        if Flow sample application name minor protocol $\neq$ "AmazonAWS", "AmazonVideo", "Azure", "Twitch", "Facebook", or "YouTube" **then**
            Delete the flow sample
        end if
    end for
    Save all the remaining samples
    Create a new video quality category column
    Read the updated flow sample label
end for
Save each updated CSV file
Output YVAIL, YVATL, YVALL, YVAP, FVAIL, FVATL, FVALL, FVAD, TVAIL, TVATL, TVALL, TVAP dataset

3.3. CICFlowmeter Preprocessing

CICFlowmeter preprocessing is another method for traffic metering and flow exportation. However, this tool lacks an integrated nDPI library function, leading to uninspected flow samples that may include a large number of flows generated by other software applications. To address this, it is necessary to use the Wireshark network analyzer tool. Wireshark's statistics conversations window displays all mutual traffic between two specific endpoints, including information from the link layer, IP layer, and transport layer. Typically, link layer traffic consists of Ethernet or IEEE 802.11 standard traffic [29], the IP layer covers both IPv4 and IPv6 traffic, and the transport layer includes TCP and UDP traffic.

Since the MAC address of the link layer is uniquely determined before packet sniffing, the next step is to sort all conversations in the IP layer by the number of packets, from largest to smallest. Video traffic is generally characterized by a high packet count and large byte size, and web videos delivered via CDNs may involve multiple IP addresses. By comparing the start time and duration of each conversation in seconds, these features can help identify the most prominent one to three conversations.

After selecting the most relevant one to three conversations from the Wireshark IP layer conversation window, these selected packets are then exported as a new PCAP file with the same video attribute category. Using CICFlowmeter, flow samples for each new video attribute category are exported, and a column is labeled with the corresponding name. All flow samples with zero-second durations, which have no practical meaning, are deleted, and the remaining flow samples for each video attribute category are saved. Finally, all video-category-attribute flow samples are merged into a single dataset, and this process is repeated for each of the three web video platform datasets, as outlined in the CICFlowmeter Video Quality Category (CICQC) Algorithm 3.

Algorithm 3 Pseudo-code for CICFlowmeter video Quality Category dataset generation (CICQC)

```
Input: Each PCAP file from the video attribute category PCAP files
Output: TVQC, YVQC, FVQC dataset
for Each PCAP file in the video attribute category PCAP files do
    Filter the most prominent one to three conversations in the Wireshark conversion module
    Save filtered packets as a new same video attribute category PCAP file
end for
Save all updated PCAP files
for Each PCAP file in the updated PCAP files do
    CICFlowmeter exports flow samples from each PCAP file
    Create a new column and label with the same video attribute category name for each flow sample
    Save all flow samples as a dataset
end for
Save three datasets according to video attribute categories
for Each dataset in datasets do
    for Each flow sample in the dataset do
        if Flow duration = 0 then
            Delete the flow sample
        end if
    end for
    Save all remaining samples
end for
Update all three datasets
Output TVQC, YVQC, FVQC dataset
```

4. Analysis of Proprietary Internet Video Traffic Datasets

For the data preprocessing described in the previous section, we used the nDACPC algorithm to generate six application and protocol category datasets: nILAD, nTLAD, nPAD, nLLAD, nILPD, nTLPD, nLLPD, and nPPD. Additionally, we leveraged the nYFTQC and CICQC algorithms to generate a total of fifteen proprietary video quality datasets: YVAIL, YVATL, YVALL, YVAP, FVAIL, FVATL, FVALL, FVAD, TVAIL, TVATL, TVALL, TVAP, TVQC, YVQC, and FVQC. This section analyzes the generated dataset characteristics from three perspectives: the imbalance ratio, flow sample histogram density and cumulative distribution, and flow-level feature correlations.

Figure 3 shows the flow record counts and distributions for YouTube, Facebook, and Twitch video quality category proprietary datasets. Across all datasets, the total number of flow records in the VLS mode consistently exceeds that in the VoD mode under identical resolution and frame rate conditions. Contrary to expectations, higher resolution does not necessarily result in more traffic or flow samples. Video traffic consumption is influenced by various video-related factors, and the number of flow samples reflects these complexities. For the nDPI video quality category datasets, all classes exhibit similar flow record counts within each category. However, Twitch generates more flow records in total and has higher sample volumes per category compared to YouTube and Facebook. In contrast, the CICFlowmeter video quality category datasets show that YouTube produces the most flow records overall. There is significant variation in sample size between categories; for instance, the 720P_30FPS_live category has 13 times more flow samples than the 720P_30FPS_VoD category. Additionally, CICFlowmeter datasets generally have category sample sizes five times larger than the corresponding nDPI categories in VLS mode. In VoD mode, the CICFlowmeter tool generates more flow samples for YouTube and Facebook compared to NFStream. However, on the Twitch platform, its sample volume is less than three times that of the corresponding NFStream dataset. Finally, the sample counts and distributions remain consistent across all accounting modes in the video quality proprietary datasets.

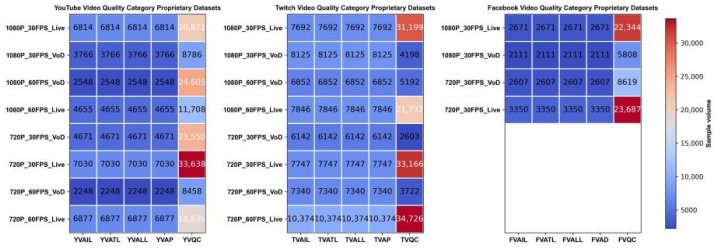


Figure 3. Video quality category proprietary datasets sample distribution.

It is worth noting that the proportions of applications and categories vary significantly after sequencing the selected video-attribute datasets. Table 1 presents the top application and protocol categories by flow sample count across different web video platforms. For the nDPI Facebook and YouTube video quality datasets, unknown flows constitute the majority, accounting for over 95% and 71% of total samples, respectively. TLS and QUIC are the two largest categories of parsed protocol categories. For the Twitch platform, TLS accounts for over 80% of the total flow volume, reflecting the trend of popular streaming platforms adopting HTTP/2 and HTTP/3 standards [30,31]. These standards integrate advanced encryption protocols, making detection and classification increasingly challenging. The flow samples of application categories exhibit varying distribution patterns. In the Facebook and YouTube datasets, the unspecified category represents the highest proportion. However, Facebook traffic is limited to the SocialNetwork category, while YouTube traffic includes Web and Media categories, highlighting the nDPI library's limitations in parsing Facebook platform traffic. For Twitch, the Web category dominates flow volume, followed by Unspecified, Video, and Cloud applications. Table 1 also lists frequently decoded server names based on SNI, including facebook.com, twitch.tv, and youtube.com. While the total number of decoded server names is relatively small, they exhibit distinct characteristics. These request servers can be categorized into direct video streaming URLs, CDN accelerator cache

domains, DNS resolution addresses, API development interfaces, gateway addresses, user login encryption addresses, and chat and conversation component addresses. Despite their diverse functions, all these addresses are related to video flow samples, further enhancing the confidence, explainability, and reliability of the datasets.

Table 1. Top represented application, protocol categories, and requested server names with quantities in three video quality category proprietary datasets.

Dataset	Top Represented Application Categories	Sample Count	Top Represented Protocol Categories	Sample Count	Top Represented Requested Server Names
nDPI Facebook Video Quality Category Proprietary Dataset	Unspecified	41,156	Unknown	41,156	www.facebook.com
	SocialNetwork	1800	QUIC	1696	scontent.xx.fbcdn.net
			TLS	104	gateway.facebook.com
nDPI Twitch Video Quality Category Proprietary Dataset	Web	198,352	TLS	202,108	www.twitch.tv
	Unspecified	46,364	Unknown	46,364	static.twitchcdn.net
	Video	2696			video-weaver.fra05.hls.ttvnw.net
	Cloud	1052			static-cdn.jtvnw.net
	Media	8			gql.twitch.tv
nDPI YouTube Video Quality Category Proprietary Dataset	Unspecified	109,856	Unknown	109,856	www.youtube.com
	Web	41,272	TLS	41,180	r1—sn-5go7yne6.googlevideo.com
	Media	3308	QUIC	3400	signaler-pa.youtube.com

4.1. Imbalance Ratio

To assess the distribution balance of samples across different label categories in the proprietary Internet video traffic datasets, suppose all classes $x_1, x_2, \ldots, x_n$ in the dataset are arranged in ascending order of magnitude and divided into quartiles, with the quartiles being the lower quartile for values in the 25% position and the upper quartile for values in the 75% position. Equation (2) shows the generic quartile estimation formula that allows the first- and third-quartile values to be calculated. Equation (1) gives the dimensionless quartile coefficient of the dispersion (cvq) calculation method based on Equation (2), where p can be either the lower or upper quartile with values of 0.25 and 0.75, respectively, n is the number of all classes in the dataset, k is the location index of a single class, and α is a coefficient determined by p and n. Symbol $\lfloor \cdot \rfloor$ indicates a round downward integer operation. Another dimensionless quantity evaluation metric is the coefficient of variation (cv), which can also be used to measure the degree of dispersion. It is the ratio of the standard deviation and the mean of the number of samples from all classes as shown in Equation (5). We use the following metrics to measure the imbalance rate in the proprietary Internet video traffic datasets.

$$cvq = \frac{q(0.75) - q(0.25)}{q(0.75) + q(0.25)} \tag{1}$$

$$q(p) = x_{(k)} + \alpha(x_{(k+1)} - x_{(k)}) \tag{2}$$

$$\alpha = p(n+1) - \lfloor p(n+1) \rfloor \tag{3}$$

$$k = \lfloor p(n+1) \rfloor \tag{4}$$

$$cv = \frac{\sqrt{n\sum_{i=1}^{n}\left(x_i - \frac{\sum_{i=1}^{n} x_i}{n}\right)^2}}{\sum_{i=1}^{n} x_i} \tag{5}$$

As shown in Table 2, Application category datasets and Protocol category datasets exhibit the highest dimensionless quartile coefficient and coefficient of variation among all datasets. This is attributed to the substantial differences in the number of samples across various labels, as illustrated in Figures 2 and 3. The nDPI Twitch video quality category proprietary datasets exhibit the lowest cvq and cv coefficients among all datasets due to bal-

anced sample volume distribution across categories. In contrast, the corresponding TVQC dataset has *cvq* and *cv* coefficients that are over 11 times and 5 times higher, respectively. For YouTube and Facebook video quality datasets, the *cvq* coefficient is approximately 0.5, indicating a medium–high value. However, the *cv* coefficient for Facebook is more than twice that of YouTube. Overall, the *cvq* and *cv* coefficients in CICFlowmeter video quality datasets are slightly higher than those in the nDPI datasets. Table 2 also presents some datasets' imbalanced rates performance of categories in existing works. Compared to other flow-level datasets, the variation in flow sample counts between categories depends on dataset design. While some datasets achieve perfect balance with *cvq* and *cv* coefficients of 0, our dataset maintains a generally low imbalance rate despite the inherent differences in category volumes.

Table 2. Measurement of the categories' imbalance rate in the proprietary Internet video traffic datasets.

Dataset	***cvq***	***cv***	**Dataset**	***cvq***	***cv***	**Dataset**	***cvq***	***cv***
nILAD	0.9620	2.9209	YVAIL	0.4127	0.3996	TVAIL	0.0719	0.1585
nTLAD	0.9620	2.9209	YVATL	0.4127	0.3996	TVATL	0.0719	0.1585
nPAD	0.9620	2.9209	YVALL	0.4127	0.3996	TVALL	0.0719	0.1585
nLLAD	0.9620	2.9209	YVAP	0.4127	0.3996	TVAP	0.0719	0.1585
nILPD	0.9713	2.9478	FVAIL	0.4983	0.8448	TVQC	0.7896	0.8538
nTLPD	0.9713	2.9478	FVATL	0.4983	0.8448	YVQC	0.4738	0.5037
nLLPD	0.9713	2.9478	FVALL	0.4983	0.8448	FVQC	0.5639	0.6094
nPPD	0.9713	2.9478	FVAD	0.4983	0.8448			
Reference	*cvq*	*cv*	Reference	*cvq*	*cv*	Reference	*cvq*	*cv*
[10]	0.3522	0.3881	[11]	0.8321	1.0149	[12]	0.0	0.0
[13]	0.5534	0.9055	[14]	0.3740	1.2235	[16]	0.0	0.0
[20]	0.4365	1.2184	[21]	0.5	0.4082	[25]	0.9669	2.3018

4.2. Video Traffic Histogram Density

In this section, we analyze the video traffic histogram density and distribution across three key features, bidirectional flow duration, bidirectional packets, and bidirectional bytes, for all video quality category proprietary datasets. To analyze the probability density function (PDF) and cumulative density function (CDF) of a specific video traffic feature vector, a histogram is used as an approximation of the PDF. Each bin represents an interval of the variable, with the normalized height approximating the probability density in that range. Increasing the number of bins while reducing bin width enhances the histogram's accuracy, allowing it to converge to the true PDF. To normalize the histogram, frequencies are scaled so the total area under the histogram equals 1, as defined in Equation (6).

$$p_i = \frac{f_i}{mh} \tag{6}$$

where

- f_i is the frequency of points in the i-th bin of the histogram;
- p_i is the height of the i-th bin of the histogram;
- m is the total number of data points;
- h is the bin width.

In practice, bin width is determined using the Sturges and Freedman–Diaconis rules, selecting the strategy that minimizes bin width. The cumulative density function (CDF) is derived from the PDF. The CDF can be estimated from a histogram-based PDF by summing the normalized heights of all bins up to and including the bin corresponding to a given value. For the bin containing the value, a proportional contribution is added based on the position of the value within the bin. This cumulative sum provides an estimate of the cumulative probability distribution. Figure 4 shows the traffic flow duration histogram

density and cumulative distribution for proprietary datasets across all video quality categories. It reveals that for most nDPI datasets from all three platforms, the majority of flow durations concentrate in the first 25 s, while the CICFlowmeter datasets exhibit the opposite trend, with most flow durations concentrated in the range of 125–180 s. We set the flow metering timer to 180 s, which means the maximum flow duration in the samples will be capped at 180 s. The two DFI tools employ different flow metering strategies. Figure 5 displays the traffic flow bidirectional packet accumulation histogram density and cumulative distribution for all datasets. It shows that most bidirectional packet accumulations fall within the 0–1500 range. However, the nDPI Twitch dataset stands out with a significant concentration of flow samples having very large bidirectional packet accumulations, particularly in the range of 120,000–150,000. Figure 6 presents the traffic flow bidirectional byte accumulation histogram density and cumulative distribution for all datasets. The majority of bidirectional byte accumulations are concentrated within the first 175,000 bytes across all datasets. Similarly, CICFlowmeter datasets consistently show higher byte accumulations than nDPI datasets.

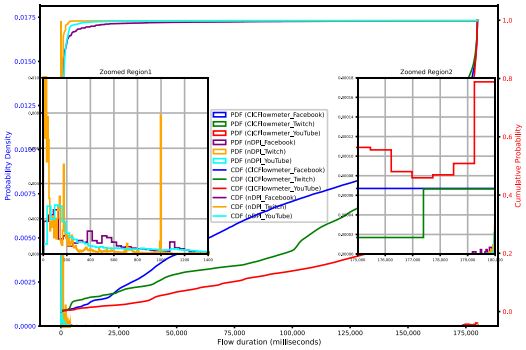


Figure 4. Traffic flow duration histogram density and cumulative distribution.

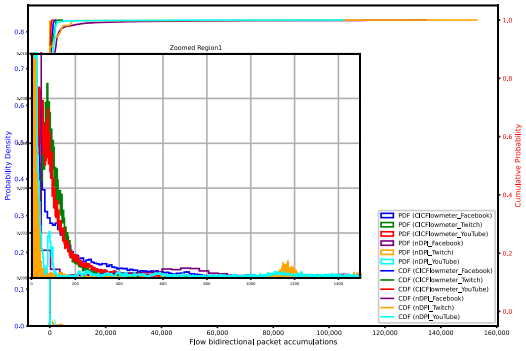


Figure 5. Traffic flow bidirectional packet accumulation histogram density and cumulative distribution.

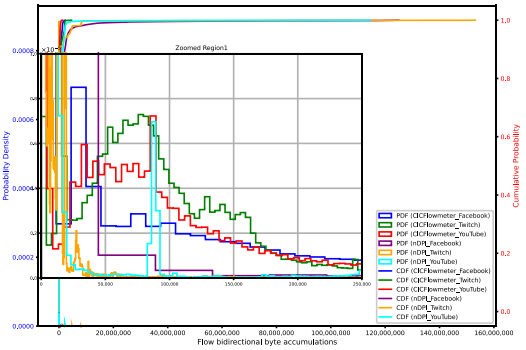


Figure 6. Traffic flow bidirectional byte accumulation histogram density and cumulative distribution.

4.3. Feature Correlation

Correlation analysis between different features in proprietary Internet video traffic datasets is essential. Based on different DFI tool labels, we merged the datasets into two categories as shown in Figure 3, and selected all numerical features for Pearson's Correlation Coefficient computation. The number of numerical features selected under the CICFlowmeter and nDPI tools were 80 and 68, respectively. Given a set of vectors $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$, where each $V_i \in \mathbb{R}^n$, m represents the total number of features. Each pair of vectors (V_i, V_j) with $i \neq j$ is chosen to compute the Pearson's Correlation Coefficient ρ_{V_i,V_j}. The formula for the Pearson's Correlation Coefficient between two vectors $X = V_i$ and $Y = V_j$ is defined in Equation (7).

$$\rho_{V_i,V_j} = \frac{\sum_{k=1}^{n}(V_{i_k} - \bar{V}_i)(V_{j_k} - \bar{V}_j)}{\sqrt{\sum_{k=1}^{n}(V_{i_k} - \bar{V}_i)^2}\sqrt{\sum_{k=1}^{n}(V_{j_k} - \bar{V}_j)^2}} \tag{7}$$

where

- V_{i_k} and V_{j_k} are the k-th elements of vectors V_i and V_j;
- $\bar{V}_i$ and $\bar{V}_j$ are the means of vectors V_i and V_j;
- n is the number of elements in each vector.

Based on the computational results presented in Figure 7, we observe that some features have a value of 0 in all samples. These concentrate on specific TCP flags, unidirectional flow in a certain direction, and some flow-related temporal features, which must be filtered using feature selection in the training model. The Pearson's Correlation Coefficient for most features is around 0, indicating a low correlation between them. In contrast to Figure 7, Figure 8 shows fewer features with a value of 0. However, the nDPI tool employs numerous statistical methods to generate features, which leads to strong correlations among them, primarily focusing on features related to packet interarrival time and packet size. This increases noise and complexity in the training model, necessitating additional feature selection and dimensionality reduction techniques to mitigate the impact of high correlation.

Figure 7. Feature correlation matrix in video quality category datasets created with CICFlowmeter.

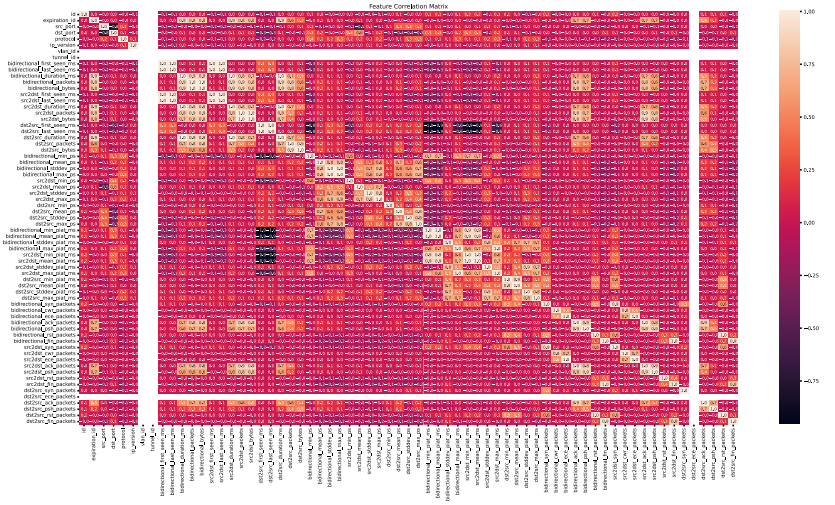


Figure 8. Feature correlation matrix in video quality category datasets created with nDPI.

5. Conclusions

This paper introduces a novel proprietary Internet video traffic dataset generation algorithm (nYFTQC), and we comprehensively demonstrate the fine-grained dataset generation process using various DFI and network traffic analysis tools. The steps involved, including data collection, collation, and label selection for each sample in the nDPI video quality category datasets, are fully transparent. All samples in the datasets are highly interpretable through analyses of imbalance rates, histogram densities, distributions, and feature correlations. Although the imbalance rate of samples across different labels is low, their feature correlation coefficients are high, accompanied by low values in duration, packets, and byte distributions. This allows for the synthesis and processing of multiple individual datasets of the same type according to training model requirements. Additionally, we developed the CICQC algorithm and generated the corresponding CICFlowmeter video quality category datasets. While these datasets contain more samples, their interpretability is limited due to the lack of metadata, and they exhibit a high imbalance rate across different labels; however, the advantage is that the feature correlation coefficient is low. The datasets generated in this study provide valuable insights for the analysis of Internet traffic in the video category.

Author Contributions: Conceptualization, M.-D.C., T.C. and E.G.; methodology, T.C.; software, T.C.; validation, E.G., A.I. and M.-D.C.; formal analysis, T.C.; investigation, T.C.; resources, T.C.; data curation, T.C.; writing—original draft preparation, T.C.; writing—review and editing, T.C., E.G., A.I. and M.-D.C.; visualization, T.C.; supervision, M.-D.C.; project administration, A.I. and E.G.; funding acquisition, M.-D.C. and T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the European Social Fund within the Project No 8.2.2.0/20/I/008 «Strengthening of PhD students and academic personnel of Riga Technical University and BA School of Business and Finance in the strategic fields of specialization» of the Specific Objective 8.2.2 «To Strengthen Academic Staff of Higher Education Institutions in Strategic Specialization Areas» of the Operational Programme «Growth and Employment», and also supported by grant PID2023-148214OB-C21 funded by MICIU/AEI/10.13039/501100011033 and FEDER/EU. This research is being carried out within the framework of the Recovery, Transformation and Resilience Plan funds, financed by the European Union (Next Generation).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets are publicly available at https://github.com/dossos/Datasets (accessed on 2 January 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kwak, K.T.; Lee, S.Y.; Ham, M.; Lee, S.W. The Effects of Internet Proliferation on Search Engine and Over-the-Top Service Markets. *Telecommun. Policy* **2021**, *45*, 102146.
2. Li, W.; Canini, M.; Moore, A.W.; Bolla, R. Efficient Application Identification and the Temporal and Spatial Stability of Classification Schema. *Comput. Netw.* **2009**, *53*, 790–809.
3. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of Encrypted and VPN Traffic Using Time-Related Features. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy, Rome, Italy, 19–21 February 2016; SCITEPRESS—Science and Technology Publications: Rome, Italy, 2016; pp. 407–414.
4. Habibi Lashkari, A.; Draper Gil, G.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of Tor Traffic Using Time Based Features. In Proceedings of the 3rd International Conference on Information Systems Security and Privacy, Porto, Portugal, 19–21 February 2017; SCITEPRESS—Science and Technology Publications: Porto, Portugal, 2017; pp. 253–262.

5. Habibi Lashkari, A.; Kaur, G.; Rahali, A. DIDarknet: A Contemporary Approach to Detect and Characterize the Darknet Traffic Using Deep Image Learning. In Proceedings of the 2020 the 10th International Conference on Communication and Network Security, Tokyo, Japan, 27 November 2020; pp. 1–13.
6. Bader, O.; Lichy, A.; Dvir, A.; Dubin, R.; Hajaj, C. OSF-EIMTC: An Open-Source Framework for Standardized Encrypted Internet Traffic Classification. *Comput. Commun.* **2024**, *213*, 271–284.
7. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.
8. Muehlstein, J.; Zion, Y.; Bahumi, M.; Kirshenboim, I.; Dubin, R.; Dvir, A.; Pele, O. Analyzing HTTPS Encrypted Traffic to Identify User's Operating System, Browser and Application. In Proceedings of the 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2017; pp. 1–6.
9. Pham, T.-D.; Ho, T.-L.; Truong-Huu, T.; Cao, T.-D.; Truong, H.-L. MAppGraph: Mobile-App Classification on Encrypted Network Traffic Using Deep Graph Convolution Neural Networks. In Proceedings of the Annual Computer Security Applications Conference, Virtual Event, 6 December 2021; pp. 1025–1038.
10. Liu, L.; Yu, Y.; Wu, Y.; Hui, Z.; Lin, J.; Hu, J. Method for Multi-Task Learning Fusion Network Traffic Classification to Address Small Sample Labels. *Sci. Rep.* **2024**, *14*, 2518
11. Salman, O.; Elhajj, I.H.; Chehab, A.; Kayssi, A. Towards Efficient Real-Time Traffic Classifier: A Confidence Measure with Ensemble Deep Learning. *Comput. Netw.* **2022**, *204*, 108684.
12. Arestrom, E.; Carlsson, N. Early Online Classification of Encrypted Traffic Streams Using Multi-Fractal Features. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April –2 May 2019; pp. 84–89.
13. Wu, H.; Wang, L.; Cheng, G.; Hu, X. Mobile Application Encryption Traffic Classification Based On TLS Flow Sequence Network. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
14. Xiao, X.; Xiao, W.; Li, R.; Luo, X.; Zheng, H.-T.; Xia, S.-T. EBSNN: Extended Byte Segment Neural Network for Network Traffic Classification. *IEEE Trans. Dependable Secur. Comput.* **2021**, *1*, 1.
15. Gabilondo, Á.; Fernández, Z.; Viola, R.; Martín, Á.; Zorrilla, M.; Angueira, P.; Montalbán, J. Traffic Classification for Network Slicing in Mobile Networks. *Electronics* **2022**, *11*, 1097.
16. Wang, Z.; Dong, Y.; Shi, H.; Yang, L.; Tang, P. Internet Video Traffic Classification Using QoS Features. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), Kauai, HI, USA, 15–18 February 2016; pp. 1–5.
17. Dong, Y.; Zhao, J.; Jin, J. Novel Feature Selection and Classification of Internet Video Traffic Based on a Hierarchical Scheme. *Comput. Netw.* **2017**, *119*, 102–111.
18. Canovas, A.; Jimenez, J.M.; Romero, O.; Lloret, J. Multimedia Data Flow Traffic Classification Using Intelligent Models Based on Traffic Patterns. *IEEE Netw.* **2018**, *32*, 100–107.
19. Hayashi, K.; Ooka, R.; Miyoshi, T.; Yamazaki, T. P2PTV Traffic Classification and Its Characteristic Analysis Using Machine Learning. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 18–20 September 2019; pp. 1–6.
20. Costa Da Silva, L.; Monks, E.; Yamin, A.; Reiser, R.; Bedregal, B. ω-IvE Methodology: Admissible Interleaving Entropy Methods Applied to Video Streaming Traffic Classification. *Int. J. Approx. Reason.* **2024**, *164*, 109061.
21. Wu, Z.; Dong, Y.; Jin, J.; Wei, H.-L.; Xie, G. Multimedia Traffic Classification for Imbalanced Environment. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 1838–1852.
22. Wu, H.; Li, X.; Cheng, G.; Hu, X. Monitoring Video Resolution of Adaptive Encrypted Video Traffic Based on HTTP/2 Features. In Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, 10 May 2021; pp. 1–6.
23. Bukhari, S.M.A.H.; Afaq, M.; Song, W.-C. PPS: A Packets Pattern-Based Video Identification in Encrypted Network Traffic. In Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, Taormina, Messina, Italy, 4 December 2023; pp. 1–6.
24. Ozkan, H.; Temelli, R.; Gurbuz, O.; Koksal, O.K.; Ipekoren, A.K.; Canbal, F.; Karahan, B.D.; Kuran, M.Ş. Multimedia Traffic Classification with Mixture of Markov Components. *Ad Hoc Netw.* **2021**, *121*, 102608.
25. Alcock, S.; Nelson, R. Measuring the Accuracy of Open-Source Payload-Based Traffic Classifiers Using Popular Internet Applications. In Proceedings of the 38th Annual IEEE Conference on Local Computer Networks—Workshops, Sydney, Australia, 21–24 October 2013; pp. 956–963.
26. Rescio, T.; Favale, T.; Soro, F.; Mellia, M.; Drago, I. DPI Solutions in Practice: Benchmark and Comparison. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021; pp. 37–42.

27. Deri, L.; Martinelli, M.; Bujlow, T.; Cardigliano, A. nDPI: Open-Source High-Speed Deep Packet Inspection. In Proceedings of the 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), Nicosia, Cyprus, 4–8 August 2014; pp. 617–622.
28. Aouini, Z.; Pekar, A. NFStream A Flexible Network Data Analysis Framework. *Comput. Netw.* **2022**, *204*, 108719.
29. *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*; IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN. IEEE: Piscataway, NJ, USA, 2021; pp. 1–767.
30. Thomson, M.; Benfield, C. HTTP/2; RFC Editor, 2022; p. RFC9113. Available online: https://www.rfc-editor.org/rfc/rfc9113.html (accessed on 2 December 2024).
31. Bishop, M. HTTP/3; RFC Editor, 2022; p. RFC9114. Available online: https://www.rfc-editor.org/rfc/rfc9114.html (accessed on 2 December 2024).

[Publication 4]: Bitrate-based Video Traffic Classification

Chen, T., Grabs, E., Ipatovs, A., Pētersons, E., Ancāns, A. Bitrate-based Video Traffic Classification. In: 2023 PhotonIcs & Electromagnetics Research Symposium (PIERS 2023): Proceedings, Czech Republic, Prague, 3-6 July, 2023.

Bitrate-based Video Traffic Classification

Tianhua Chen, Elans Grabs, Aleksandrs Ipatovs, Ernests Pētersons, and Arnis Ancāns
Institute of Telecommunications, Riga Technical University, Azenes street 12, LV-1048, Riga, Latvia

Abstract— Granular network traffic classification is gaining high priority, which is crucial for the Internet Service Providers (ISP), Over The Top (OTT) providers' operation and maintenance management and users' Quality of Experience (QoE) improvement. Streaming video category traffic takes up a significant proportion of Internet traffic. Its ground truth sources include the application types, streaming network communication protocols, resolution, refresh rates, video encoding protocols, physical network resource bandwidth, and associated with the video source. However, the credibility of the traffic classification work based on the ground truths above-mentioned is not high since the quality of the video source cannot be guaranteed. The user's perception is poor even when watching higher resolution and refresh rate in a particular scenario. Secondly, different video platforms use different technical standards, which will inevitably cause video quality compression loss in transmission and viewing. The user viewing experience varies greatly, even under the same standard. In this paper, we propose to implement the traffic classification task by calculating the video bitrates of Video on Demand (VoD) and Video Live Streaming (VLS) as accurate classification labels and using machine learning techniques, in which we will examine the real-time bitrates during real-world video transmission compared with the bitrates set by theoretical recommendations to find the differences between the two scenarios.

2023 Photonics & Electromagnetics Research Symposium (PIERS) | 979-8-3503-1284-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/PIERS59004.2023.10221281

1. INTRODUCTION

Internet traffic generated by video streaming is proliferating, and the perceptual experience of users watching Video on Demand (VoD) and Video Live Streaming (VLS) is improving have benefited from the development of adaptive streaming protocols. To achieve more refined Internet traffic classification, researchers have learned more a priori knowledge about streaming traffic, including video-related parameters such as resolution, refresh rate, video compression coding format, video bitrate, bitrate controller mode, streaming content categories, and so forth. Video transmission-related parameters include transmission protocols, network resource bandwidth, and ultra-low latency settings. Server-related knowledge contains streaming platforms, server throughput, Server Name Indication (SNI), etc. The knowledge aforementioned usually be considered as ground truth, and labeling records perform supervised learning for traffic classification purposes. Our previous work [1] used the resolution and refresh rate of videos that generated video quality labels and achieved 97% classification accuracy when using Convolutional Neural Networks (CNN). However, videos' resolution and refresh rates were determined based on YouTube manual selection and traffic sniffing capture under randomly selected scenarios of watching individual videos. The drawback of this approach is that it is not universal, and it remains to be studied whether a high video traffic classification performance metric can be achieved for different streaming platforms, protocols, and video source quality. In contrast, the reliability of its classification from the Internet Service Providers (ISP's) network traffic management perspective remains to be proven. To address the above issues, we propose a state-of-art streaming video traffic classification scheme based on video bitrate connected with Graph Convolutional Networks (GCN) [2].

2. VIDEO TRAFFIC DATASETS GENERATION BASED ON BITRATE

Bitrate refers to the number of bits per second representing the volume of video or audio data after compression and encoding, it also can indicate the amount of data displayed per second after the image is compressed, measured in kilobits per second. Usually, the raw filmed video needs to use compression encoders to diminish the volume of video files and ensure that the video quality is not destroyed. Standard video codecs include Advanced Video Coding (AVC), High-efficiency video coding (HEVC), AOMedia Video 1 (AV1), VP9, etc. Different encoders have bitrate control algorithms, determining how to consume the bitrate budget over time. The bitrate control techniques significantly affect compression efficiency, video quality, and speed. Commonly utilized bitrate control algorithms include Constant Bitrate (CBR), which allows the average bitrate to remain constant while sacrificing video quality; Variable Bitrate (VBR), which allows the video quality to stay consistent while the Bitrate fluctuates; and Capped VBR, which enables the bitrate to fluctuate within a specific limit or ceiling while the video quality remains constant. Bitrate

encoding time also dramatically impacts video quality; for gaming VLS, the video needs to be compressed in real-time at a high refresh rate of 50 fps or 60 fps, and the encoder usually sacrifices the video quality to increase the speed. Video resolution is also a prominent conclusive factor affecting the encoding Bitrate. The resolution designates the width of the video image multiplied by the height in pixels, and above encoders all support processing video in Full High-Definition (FHD, 1080 p) resolution format. Equation (1) presents the parameters for calculating the bitrate of a video file.

$$Bitrate = \frac{w * h * fps * bpp}{1000} \tag{1}$$

where w is the video's resolution's width in pixels, h is height is pixels, fps is the number of frames per second in the video, bpp is the total bits per pixel value used to calculate the optimal video bitrate for live streaming and is usually set to 0.1, and the unit of $Bitrate$ is kbps. In addition, the Group of Pictures (GOP) length, Image parameters, and frame type, such as intra-frame, are also crucial parameters that guide the video's bitrate. The processed video to be viewed on different streaming platforms also requires slicing or transcoding, including changing the video container format, codec, video resolution, etc. Currently, most streaming platforms use the transport paradigm of HTTP adaptive streaming (HAS), an HTTP-based communication protocol between the streaming client and HTTP servers. The evolving mainstream solutions include HTTP Live Streaming (HLS) and Moving Picture Experts Group-Dynamic Adaptive Streaming over HTTP (MPEG-DASH) protocols. MPEG-DASH is an open-source solution that divides multimedia files into multiple chunks, indexes them using Media Presentation Description (MPD) XML file, and transmits them to the client for playback using HTTP.

In this paper, we will use MPEG-DASH protocol to simulate the transfer of a single video file with a bitrate of 8000 kbps, using Nginx as the web server, dash.js as the player, and tcpdump to capture simulated video traffic. In addition, we sniff video traffic from popular real-world platforms such as Twitch, Vimeo, YouTube, Facebook, and IPTV. IPTV video traffic was captured when playing IPTV VLS in the VLC media player, and relevant video information was obtained from the VLC decoder. Other platforms' video traffic was sniffed individually by Wireshark based on watching a single video and all video play duration was around 300 seconds. Due to the HAS bitrate adaptation function, it was compulsory to select a video resolution of 1080 p frame rate 30 fps or 60 fps, and this adaptation function would not be triggered under the bandwidth resources guaranteed. Table 1 summarizes captured video traffic PCAP files statistics and recommended bitrate range for streaming platforms. We can learn that different platforms have different recommended video bitrate range with other video resolution and frame rates, with over 30% discrepancy between maximum and minimum recommended video bitrate. There is still a half percentage of difference compared to theoretically calculated values of 6220.8 kbps at 1080 p30 fps and 124416 kbps at 1080 p60fps when using Equation (1). To explore the bitrate of a single video randomly selected for playback on different platforms, it can be calculated by examining the size and duration of the VoD file, but this is limited to specific platforms such as YouTube, for other platform videos or VLS, the video file size information cannot known in advance, the average bits/s parameter of the captured PCAP file will be used to infer the bitrate of the source video.

The average bits/s parameter covers not only the video's bitrate but also the audio and some other system application traffic information, which is more significant than the raw bitrate of the video if the bandwidth resources are fully guaranteed. However, for the simulated PCAP file, the average bits/s parameter is smaller than the source video bitrate by more than 10% due to a buffer pause in the video playback process, resulting in insufficient total captured packets. Overall, among the 10 PCAP files, excluding one known video bitrate, the average bits/s of 4 of the remaining 9 PCAP files all do not reach their recommended video bitrate ranges. Still, they are all certified with the same video resolution and refresh rate. If only resolution and refresh rate labeling as ground truth and used for supervised learning classification, it not only causes the user's viewing perception to be poor but the credibility of their classification results for OTT content providers' research also be significantly reduced. Combining the various types of video bitrates analyzed above, under the condition of 1080 p resolution, the video bitrates can be divided into three categories, including Low Bitrate, which is less than 3000 kbps; Medium Bitrate, is 3000–6000 kbps. More than 6000 kbps is High Bitrate, and the above label information can also be matched with different platforms for multi-class classification. After determining the category label information, those PCAP files will be preprocessed to export flow-level samples to form video traffic datasets.

Table 1: Captured video traffic PCAP files statistics and recommended bitrate range for streaming platforms.

PCAP files	Video duration (seconds)	Packets	Bytes	Average packets/s	Average bytes/s	Average bits/s	Bitrate Range:
Twitch_1080 p60 fps_VLS	300.7	285948	341057199	950.7	1133 k	9071 k	> 6000 kbps
Twitch_1080 p60 fps_VoD	302.1	338704	378404795	1121.2	1252 k	10240 k	> 6000 kbps
Vimeo_1080 p30 fps_VoD	301.6	115295	144579406	382.3	479 k	3835 k	4000–5000 kbps
IPTV_1080 p30 fps_VLS	301.9	65673	77827618	217.6	257 k	2062 k	/
YouTube_1080 p30 fps_VLS	302	82151	90427159	272	299 k	2395 k	3000–6000 kbps
YouTube_1080 p60 fps_VLS	302.1	121712	135372706	402.8	448 k	3584 k	4500–9000 kbps
YouTube_1080 p60 fps_VoD	302.2	137917	157864565	456.4	522 k	4179 k	4500–9000 kbps
Facebook_1080 p30 fps_VLS	301.2	223363	244700154	741.5	812 k	6498 k	3000–6000 kbps
Facebook_1080 p30 fps_VoD	300	196480	219075890	654.8	730 k	5840 k	3000–6000 kbps
DASH_1080 p30 fps_VoD	298.9	133394	266851729	446.3	892 k	7142 k	= 8000 kbps

3. EXPERIMENT SETUP

Figure 1 exhaustively shows the video traffic classification flow diagram based on bitrate, and it is divided into parts (a), (b), and (c), already introduced in the last section. We will use the NFStream [3] tool to produce flow-level traffic samples and set the flow active timeout parameter as one second, which means the original flow will stop counting and generate a new flow when the bidirectional duration is over one second in part (d), and need to delete irrelevant flow samples for sample cleaning and feature selection. NFStream offers over 80 features and around 30 features elected for training, including flow bidirectional, uni-directional time durations, packets, bytes, and their statistics such as maximum, minimum, standard deviation, mean, etc. According to the bitrate label-matching situation, sample imbalance in different categories also need further over-sampling of all classes except the majority using Synthetic Minority Over-sampling Technique (SMOTE) and cleaning some samples close to the judgment boundary using Edited Nearest Neighbours (ENN) [4]. We still need to eliminate the dimensionality of features by calculating the standard score before the graph generation. A graph consists of nodes and edges, nodes represented by video traffic samples, edges decided by the calculation of pairwise Pearson correlation coefficient between the nodes, and the next step will find two nearest neighbors for each node based on the correlation matrix, the new edge contains source and target information, which will be used for index nodes and their features. After producing the graph, we set two layers of GCN and a layer size of 256 combined with the ReLU activation function, Adam optimizer, category cross-entropy loss, and learning rate assigned as 0.001. The output will have eight categories using the Softmax activation function, and the traffic classification task will convert to graph nodes classification [5]. To better validate the actual training classification effect, we divided the updated video traffic dataset into the training, validation, and testing dataset with allocation ratios of 0.35, 0.3, and 0.3, respectively, and training epochs of 300 rounds.

4. EXPERIMENT RESULT DISCUSSION

We will choose accuracy, precision, recall, and f1-score as classification performance evaluation metrics. All metrics except accuracy will calculate the macro average in the multi-class classification task, which means each category metric will be calculated separately and then obtain the arithmetic average. Figures 2 and 3 reveal training and validation datasets training loss and accuracy with epochs, respectively. As the number of epochs increases, the training loss in the training and validation datasets decreases rapidly, and the accuracy rate increase expeditiously. After around 150 epochs, the decrease in training loss and the increase in accuracy rate are all no longer marked, and the accuracy rate fluctuates up and down repeatedly. The accuracy of the training and validation dataset is 0.9 and 0.86, respectively. The metrics of the testing dataset also perform well, with a classification accuracy of 0.86, precision, recall, and f1-score of 0.83. In addition to the overall classification metrics, we also focus on the classification accuracy of each category sample, Figure 4 shows the confusion matrix of each category, and the IPTV_Low_Bitrate category gets the worst

classification result with an accuracy of 0.69. Despite the sample resampling, the distribution of the number of samples between different categories is still large, and the similarity between medium and low bitrate samples is high, making it difficult to classify them.

To further investigate the impact of different bitrates on the classification results, we remove the a priori knowledge of the streaming platform and emulation protocols as training labels and only keep bitrate-related categories of High_Bitrate, Medium_Bitrate, and Low_Bitrate. The accuracy of the testing dataset is 0.92, and the rest of the metrics, including precision, recall, and f1-score are 0.89 for three categories. The overall classification performance is better than the eight categories' dataset. Based on the confusion matrix shown in Figure 5, we can learn that the classification accuracy of the Medium_Bitrate category is the lowest at 0.82, while the classification accuracy of all other categories is over 0.92, which is excellent. After data resampling, the Medium_Bitrate category has 30% fewer samples than the Low_Bitrate category combined with Figure 7 using t-distributed stochastic neighbor embedding (t-SNE) [6] for GCN embeddings visualization, the green and brown samples representing both categories are highly overlapping. Therefore, additional samples need to be trained to improve the classification metrics. For the visualization of GCN embeddings shown in Figure 6, samples of different categories not only overlap and mix, but also have large discrete differences between samples, and further dataset cleaning is required.

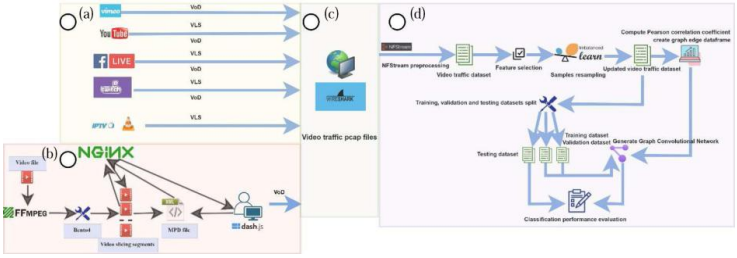


Figure 1: Bitrate-based video traffic classification flow diagram, (a) real-world VLS and VoD video streaming playback; (b) simulated streaming video playing based on DASH protocol; (c) capture real-world and simulated streaming video network traffic via Wireshark and tcpdump; (d) NFStream for export flow-level traffic samples and generate video traffic dataset, feature selection, dataset cleaning, and imbalanced categories samples resampling update raw video traffic datasets, using GCN classify video traffic category.

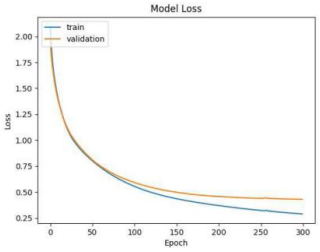


Figure 2: Training and validation loss with epochs eight categories.

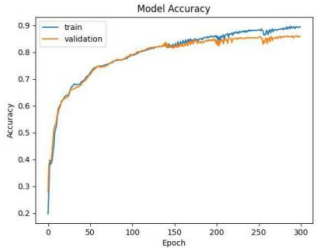


Figure 3: Training and validation accuracy with for epochs for eight categories.

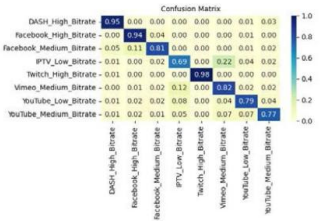


Figure 4: Confusion matrix epochs for eight categories.

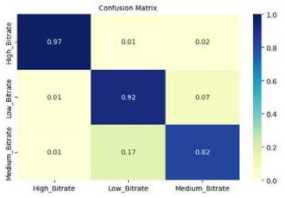


Figure 5: Confusion matrix for three categories.

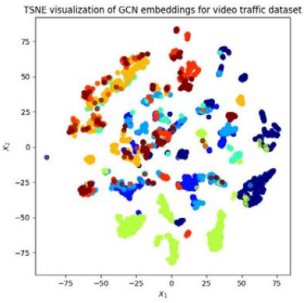


Figure 6: Visualization of GCN embeddings for eight categories of the video traffic dataset.

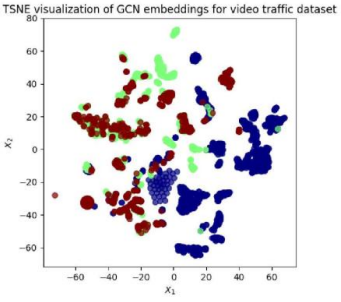


Figure 7: Visualization of GCN embeddings for three categories of the video traffic dataset.

5. CONCLUSION

This article elaborates on using video bitrate as a ground truth label for simulated and real-world video traffic classification at 1080 p resolution supervised learning classification and obtains high classification accuracy. When using GCN to create graphs and using the adjacency matrix and degree matrix of nodes has a significant improvement for feature extraction of nodes associated with the drawback is the short length time of captured traffic and the small number of derived stream samples, which also includes the problem of the unbalanced total number of samples between different categories. The solution consists of increasing the video's playing duration and refining PCAP files to obtain more network traffic samples. From the simulated video traffic PCAP file, we can find that even if the flow is segmented within one second, the data packets covered in a single flow sample are more than thousands. Therefore, a single PCAP file can still be segmented according to packet volume in the pre-processing stage to obtain more training samples. In addition, the range we set in the Medium_Bitrate category is 3000–6000 kbps, and the classification effect is unsatisfactory. Further study is needed to determine whether the category range should be further divided.

ACKNOWLEDGMENT

This work has been supported by the European Social Fund within the Project No 8.2.2.0/20/I/008 "Strengthening of PhD students and academic personnel of Riga Technical University and BA

School of Business and Finance in the strategic fields of specialization" of the Specific Objective 8.2.2 "To Strengthen Academic Staff of Higher Education Institutions in Strategic Specialization Areas" of the Operational Programme "Growth and Employment".

REFERENCES

1. Chen, T., et al., "Multiclass live streaming video quality classification based on convolutional neural networks," *Autom. Control Comput. Sci.*, Vol. 56, No. 5, 455–466, Oct. 2022, doi: 10.3103/S0146411622050029.
2. Data61, C., StellarGraph Machine Learning Library, GitHub Repository, GitHub, 2018. [Online]. Available: https://github.com/stellargraph/stellargraph.
3. Aouini, Z. and A. Pekar, "NFStream," *Comput. Netw.*, Vol. 204, 108719, Feb. 2022, doi: 10.1016/j.comnet.2021.108719.
4. Lemaitre, G., F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.*, Vol. 18, No. 17, 1–5, 2017.
5. Kipf, T. N. and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, doi: 10.48550/ARXIV.1609.02907.
6. Buitinck, L., et al., "API design for machine learning software: Experiences from the scikit-learn project," 2013, doi: 10.48550/ARXIV.1309.0238.

[Publication 5]: Network Traffic Analysis for eXtended Reality Applications

Chen, T., Grabs, E., Tasic, I., Cano, M. Network Traffic Analysis for eXtended Reality Applications. In: XVI Telematics Engineering Conference (JITEL 2023), Spain, Barcelona, 8-10 November, 2023.

XVI Jornadas de Ingeniería Telemática
JITEL 2023
La Salle – Universitat Ramon Llull

Actas de las XVI Jornadas de Ingeniería Telemática (JITEL 2023), Barcelona (España), 8-10 de noviembre de 2023.

Network Traffic Analysis for eXtended Reality Applications

Tianhua Chen[1], Elans Grabs[1], Igor Tasic[2], Maria-Dolores Cano[2]
[1]Telecommunications Institute, Riga Technical University, Azenes street 12, LV-1048 Riga, Latvia
[2]Department of Information Technologies and Communication, Universidad Politécnica de Cartagena, 30202 Cartagena, Spain

tianhua.chen@rtu.lv, mdolores.Cano@upct.es

Currently, new multimedia applications are booming, especially streaming services represented by eXtended Reality (XR) and the next generation B5G/6G networks. Network traffic generated by XR and traditional mobile and PC clients is becoming more and more intertwined and complex, especially the homogeneity of Quality of Services (QoS) parameters makes the analysis of traffic critical. In this paper, we review the research done so far in traffic classification for XR services and propose the use of graph neural networks and Quality of Experience (QoE) scores to detect and classify highly similar streaming traffic as well as to predict them in further works. The goal will be not only to secure network resources and devices but also to achieve dynamic traffic classification and resource management for heterogeneous networks with a mixture of multiple network standards to meet the requirements of Self Organizing Networks (SON).

***Keywords*- VR, AR, XR, spatial computing, metaverse, traffic classification, QoS, QoE, 5G.**

I. Introduction

Nowadays, conventional multimedia applications and services are evolved into the metaverse, including Augmented Reality (AR), Virtual Reality (VR), Mixed Reality (MR), and, in general, eXtended Reality (XR) or spatial computing, compared to unitary text, audio, images, videos, and animation that are more underlining interactive resources integrated.

Head Mounted Devices (HMDs) are one of the representative tools to realize their demands and support three degrees of freedom (3DoF), including pitch, yaw, and roll, or six degrees of freedom (6DoF) subdivided into forward, backward, left, right, up, and down. High degrees of freedom support combined with oversized Field of View (FoV) remarkably enhance users' immersive experience when using HMDs to watch graphics or videos, particularly for 360 degrees videos [1] [2].

While the content and form of multimedia resources continue to evolve, the transmission requirements for next-generation networks are becoming increasingly demanding, as reflected by parameters such as end-to-end latency, jitter, throughput, burst traffic, packet loss ratio, Round Trip Times (RTT), or network delay, among others. Those parameters embedded into Quality of Service (QoS) and Quality of user Experience (QoE) models will significantly impact users' perceptions of XR applications in 5G and future B5G/6G networks [3] [4].

Self Organizing Networks (SON) [5] [6] is one of the most vital development goals of 5G, which mainly refers to Self-Configuration, Self-Optimization, and Self-Healing. Usually, Self-Configuration is linked to Self-Learning. It lets the network understand topology, realize parameter adjustments, and update nodes. Self Healing refers to maintenance assistance, including load balancing, capacity expansion, or power outage. Self Optimization connects to traffic monitoring and performance optimization, dealing with QoS and QoE parameters.

On the other hand, traffic monitoring is a transversal action that plays a top-down role in the Internet. For example, classifying and analyzing network traffic allows for more efficient load balancing of network resources. In the face of new XR applications, how to identify new types of network traffic, i.e., traffic classification, might play a key role in allocating and optimizing network resources.

In this work-in-progress paper, we review the traffic characteristics of XR applications in heterogeneous networks and reflect on the role of combining QoS/QoE levels to detect and classify different XR application applications. The rest of the paper is organized as follows. Section 2 describes the state-of-the-art research, including QoS/QoE optimization for XR traffic and QoS-based traffic classification. The proposal for traffic classification is introduced in Section 3, and finally, the paper

summarizes XR applications' traffic classification importance and future works.

II. STATE OF THE ART

The multimedia represented by XR applications have come to HMDs after the development of PC clients and mobile phone clients. Over The Top (OTT) publishers have produced the corresponding streaming content, live video streaming, and game streaming to a larger extent.

Compared with PCs or smartphones, streaming is characterized by the larger volume size of the video source and the use of ultra-high resolution (4K or 8K). Resolution is mainly determined by Pixels Per Degree (PPD), color depth, and Frames Per Second (FPS); as the most recommended resolution for HMDs monocular resolution is 4K, the left and right eyes also support frame sequential multiplexing when playing streaming, which gives the user an excellent immersion experience.

Considering the bandwidth and delay in network transmission, it also requires powerful video compression coding protocols, including H.264/H.265, VP9 to guarantee lossless content, 50FPS, 60FPS, or even 90FPS to make users feel the video playback smoother. This is also crucial for game streaming. Streaming communication protocols are also key for XR applications, such as Real-time Transport Protocol (RTP/UDP), Real-time Streaming Protocol (RTSP), Dynamic Adaptive Streaming over HTTP (DASH), HTTP Adaptive Streaming (HAS), and Apple's HTTP Live Streaming (HLS) with matching buffer configuration, bitrate adaption, and so forth parameters that need to re-evaluate their transmission performance [7].

Next, we review the research works from the related literature that address traffic classification of XR services and applications. To the authors' knowledge, there are no works related to XR traffic classification based on QoS/QoE.

A. QoS/QoE optimization for XR traffic

For novel XR applications, many studies have started to model optimized QoE for their traffic characteristics in next-generation networks. Mattia *et al.* [8] captured the traffic of three different types of VR applications on the streaming media rendering servers, found the most suitable statistical distribution by calculating the Cumulative Distribution Function (CDF), and generated a model that conforms to the XR traffic type for simulating the scheduling of XR traffic strategy.

Jing *et al.* [9] used the Dynamic time division duplex (D-TDD) technique to improve QoE for MR users. They used a Deep Q-Network (DQN) algorithm to mitigate Inter-Cell Interference (ICI) and optimized Multidimensional Resource Allocation (MRA). DQN, as one of the reinforcement learning algorithms, has achieved a good advantage in QoE-based resource allocation. Similarly, Ismail *et al.* [10] proposed the DeepEdge framework to efficiently allocate edge resources for Edge-IoT applications, which include VR/AR-related applications. They employed Deep Neural Networks (DNNs) with Edge-IoT state mapping to a joint resource allocation operation consisting of resource allocation and QoS categories to maximize QoE scores.

Following the same trend, Cristina *et al.* [11] utilized Deep Recurrent Neural Network (DRNN) based on Gated Recurrent Units (GRUs) to assist in maximizing transmitted video block quality. To reduce VR frame latency, they combined predictive FoV correlation with viewer position viewing 360 degrees HD VR video to enable proactive FoV-centric millimeter Wave (mmWave) Small cell Base Stations (SBSs) physical layer multicast transmission.

From the perspective of Network Slicing (NS), Federico *et al.* [2] collected traffic in real VR applications and investigated their temporal correlation, focusing on the Constant Bit Rate (CBR) encoding mode, which produces more predictable traffic.

The above-mentioned research tasks is mainly focused on finding a balance between traffic and QoE for multimedia XR applications, especially in 5G scenarios. Researchers used advanced algorithms such as machine learning and reinforcement learning to dynamically predict the network resource allocation logic to optimize resource utilization and QoE scores. QoS and QoE parameters are usually set from international standards such as ITU-T G.1035 [12] standard, ITU-T P.1203 [13] standard, ITU-T P.1204 [14] standard, 3GPP TR 26.929 [15] standard. They are further subdivided into Key Performance Indexes (KPI) and Key Quality Indexes (KQI). KPI is more related to QoS parameters, including RTT, burst pulse, etc. KQI for traditional mobile phone calling applications comprises service availability, call drop rate, etc. KQI for XR application includes stalling, Motion To Photon (MTP), FoV, FPS, etc [16].

These highly segmented parameters design requires intelligent network learning to adjust the service capability dynamically. From the traffic monitoring point of view, XR applications traffic in 5G, 6G, WiFi, and other network standards in its QoS and QoE parameters have strong properties relative to other applications, which could have a better basis for this traffic detection classification and guarantee.

B. QoS-based traffic classification

The following works basically use QoS parameters as criteria for the classification task. Zheng *et al.* [17] used sliding window technology to capture slices of multimedia application flows and obtain QoS-related features. However, its multimedia applications are still limited to conventional client-based Conference Video (CV), Video on Demand (VOD), Voice on Demand (Voice), Live Video (LV), HTTP Download Video (HDV), and Peer-to-Peer video (P2P).

A different approach was followed by Huang *et al.* [18], which acted according to different QoS standardization methods, comprising Integrated Services (IntServ) and Differentiated Services (DiffServ). Specifically, DiffServ contains Differentiated Services Code Point (DSCP) tags with Assured Forwarding (AF) [19] categories used to make classifications for mixed video streaming applications. They also obtained high classification metrics.

Gabilondo *et al*. [20] employed the dynamic classification of traffic types for different applications and heterogeneous flows and transmitted them through specific slices with associated 5G QoS Identifiers (5QI). In addition, the authors proposed specific radio resource-sharing configurations to determine the most appropriate traffic priority through scheduling. The slicing advantage improves the efficiency and reliability of the most critical data regarding QoS-related packet loss or jitter.

Our previous work [21] focused on extracting the traffic intensity of different video quality traffic as features from PC clients and achieved a high accuracy of 97% using CNN for supervised classification. All labels were manually marked. However, we did not test XR applications streaming video quality classification performance, and currently, QoE score metrics are mainly used for traffic prediction and resource allocation adjustment. To the authors' knowledge, there is no related work as criteria for classification. The disadvantage of QoS parameters such as DiffServ is the inability to isolate and classify the same type of streaming traffic. Network slicing technology can provide a complete end-to-end virtual network for highly similar traffic to provide different QoS/QoE guarantees, which also could offer a reliable basis for the traffic classification task.

III. Proposal for Traffic Classification

To achieve the goal of traffic classification for XR applications based on QoE scores metrics, we propose the flow chart shown in Fig. 1. This approach could be used for traffic classification and forecasting for streaming applications in heterogeneous networks.

Assuming different types of streaming application traffic in heterogeneous networks, we will detect their corresponding QoE levels, such as mean opinion score (MOS), video quality model (VQM), peak signal-to-noise ratio (PSNR), etc., which are based on the above metrics and classify the traffic according to different calculated level labels. If XR applications are below the expected level, we will optimize the metrics enhancement and reclassify the existing traffic in combination with QoS and 5G network slicing levels parameters, etc. By doing so, the system would combine iterative optimization and dynamic traffic classification to analyze the traffic characteristics of XR applications better.

We plan to simulate and experiment with a simple, smart home scenario. Although many devices could be connected to the smart home, we only focus on devices that can run streaming applications, including PCs, mobile phones, tablets, webcam monitors, HMDs, etc. Streaming and network video-related applications include live video streaming, online games, online conferences, XR applications built into HMDs, and streaming between multiple devices such as STEAM VR clients and HMDs, among others. In a heterogeneous network environment consisting of WIFI, 4G, and 5G, different applications and network standards are used alternatively, where these applications generate highly similar network traffic. The traffic generated by streaming needs to be further modeled, correlation analyzed, and useful training features extracted, and machine learning algorithms are used to achieve traffic prediction and classification.

We propose to use Graphical Neural Networks (GNN) [22] for traffic classification and prediction in this scenario. GNN can be used not only for traffic classification but also for traffic prediction regression, which has an excellent theoretical basis for analyzing end-to-end traffic relationships. We could also add Network Intrusion Detection Prevention Systems (NIDPS) to protect users' cybersecurity. In general, the final purpose of classification is to help OTT and Internet service providers better understand the proportion of traffic by various applications and to achieve the goal of SON by combining traffic volume prediction for better allocation and load balancing of physical resources of the network system.

IV. Conclusions And Future Work

It is well known that one of the main challenges for streaming media applications in Internet-generated traffic is bandwidth consumption. This fact is enlarged with the introduction and adoption of all kinds of new XR applications; how to identify, secure, classify, and forecast this traffic are some of the questions suggested by the researchers. Besides, the challenge of having limited physical resources while maintaining QoE levels in heterogeneous networks is another important constraint. Our future work will focus on dynamic traffic classification and prediction under this XR paradigm to improve network resource utilization.

Acknowledgments

This work has been supported by the European Social Fund within the Project No 8.2.2.0/20/I/008 «Strengthening of PhD students and academic personnel of Riga Technical University and BA School of Business and Finance in the strategic fields of specialization» of the Specific Objective 8.2.2 «To Strengthen Academic Staff of Higher Education Institutions in Strategic Specialization Areas» of the Operational Programme «Growth and Employment» and also funded by grant PID2020-116329GB-C22 funded by MCIN/AEI/10.13039/501100011033.

References

[1] Y. Wang *et al.*, "A Survey on Metaverse: Fundamentals, Security, and Privacy," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 1, pp. 319–352, 2023, doi: 10.1109/COMST.2022.3202047.

[2] F. Chiariotti, M. Drago, P. Testolina, M. Lecci, A. Zanella, and M. Zorzi, "Temporal Characterization and Prediction of VR Traffic: A Network Slicing Use Case," *IEEE Trans. Mob. Comput.*, pp. 1–18, 2023, doi: 10.1109/TMC.2023.3282689.

[3] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research

Problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020, doi: 10.1109/MNET.001.1900287.

[4] R. Sanchez-Iborra, M.-D. Cano, and J. Garcia-Haro, "Revisiting VoIP QoE assessment methods: are they suitable for VoLTE?," *Netw. Protoc. Algorithms*, vol. 8, no. 2, p. 39, Jul. 2016, doi: 10.5296/npa.v8i2.9123.

[5] M. Agiwal, A. Roy, and N. Saxena, "Next Generation 5G Wireless Networks: A Comprehensive Survey," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 3, pp. 1617–1655, 2016, doi: 10.1109/COMST.2016.2532458.

[6] S. Latif, F. Pervez, M. Usama, and J. Qadir, "Artificial Intelligence as an Enabler for Cognitive Self-Organizing Future Networks," 2017, doi: 10.48550/ARXIV.1702.02823.

[7] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 1, pp. 562–585, 2019, doi: 10.1109/COMST.2018.2862938.

[8] M. Lecci, M. Drago, A. Zanella, and M. Zorzi, "An Open Framework for Analyzing and Modeling XR Network Traffic," *IEEE Access*, vol. 9, pp. 129782–129795, 2021, doi: 10.1109/ACCESS.2021.3113162.

[9] J. Song, Q. Song, Y. Kang, L. Guo, and A. Jamalipour, "QoE-Driven Distributed Resource Optimization for Mixed Reality in Dynamic TDD Systems," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7294–7306, Nov. 2022, doi: 10.1109/TCOMM.2022.3208113.

[10] I. Alqerm and J. Pan, "DeepEdge: A New QoE-Based Resource Allocation Framework Using Deep Reinforcement Learning for Future Heterogeneous Edge-IoT Applications," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 4, pp. 3942–3954, Dec. 2021, doi: 10.1109/TNSM.2021.3123959.

[11] C. Perfecto, M. S. Elbamby, J. D. Ser, and M. Bennis, "Taming the Latency in Multi-User VR 360°: A QoE-Aware Deep Learning-Aided Multicast Framework," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2491–2508, Apr. 2020, doi: 10.1109/TCOMM.2020.2965527.

[12] "ITU Telecommunication Standardization Sector ITU-T Rec G 1035 Multimedia Quality of Service and performance – Generic and user-related aspects." [Online]. Available: https://www.itu.int/rec/T-REC-G.1035-202111-I

[13] "ITU Telecommunication Standardization Sector ITU-T Rec P 1203 Models and Tools for Quality Assessment of Streamed Media." [Online]. Available: https://www.itu.int/rec/T-REC-P.1203-201710-I

[14] "ITU Telecommunication Standardization Sector ITU-T Rec P 1204 Models and tools for quality assessment of streamed media." [Online]. Available: https://www.itu.int/rec/T-REC-P.1204-202001-I

[15] "3rd Generation Partnership Project Technical report 26.929 QoE parameters and metrics relevant to the Virtual Reality (VR) user experience." [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3327

[16] H. Dong and J. S. A. Lee, "The Metaverse From a Multimedia Communications Perspective," *IEEE Multimed.*, vol. 29, no. 4, pp. 123–127, Oct. 2022, doi: 10.1109/MMUL.2022.3217627.

[17] Z. Wu, Y. Dong, X. Qiu, and J. Jin, "Online multimedia traffic classification from the QoS perspective using deep learning," *Comput. Netw.*, vol. 204, p. 108716, Feb. 2022, doi: 10.1016/j.comnet.2021.108716.

[18] Y.-F. Huang, C.-B. Lin, C.-M. Chung, and C.-M. Chen, "Research on QoS Classification of Network Encrypted Traffic Behavior Based on Machine Learning," *Electronics*, vol. 10, no. 12, p. 1376, Jun. 2021, doi: 10.3390/electronics10121376.

[19] M.-D. Cano, F. Cerdán, J. García-Haro, and J. Malgosa-Sanahuja, "A New Proposal for Assuring Services in Internet," in *Proceedings of the International Conference on Internet Computing, IC 2002, Las Vegas, Nevada, USA, June 24-27, 2002*, H. R. Arabnia and Y. Mun, Eds., CSREA Press, 2002, pp. 379–384.

[20] Á. Gabilondo *et al.*, "Traffic Classification for Network Slicing in Mobile Networks," *Electronics*, vol. 11, no. 7, p. 1097, Mar. 2022, doi: 10.3390/electronics11071097.

[21] T. Chen *et al.*, "Multiclass Live Streaming Video Quality Classification Based on Convolutional Neural Networks," *Autom. Control Comput. Sci.*, vol. 56, no. 5, pp. 455–466, Oct. 2022, doi: 10.3103/S0146411622050029.

[22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

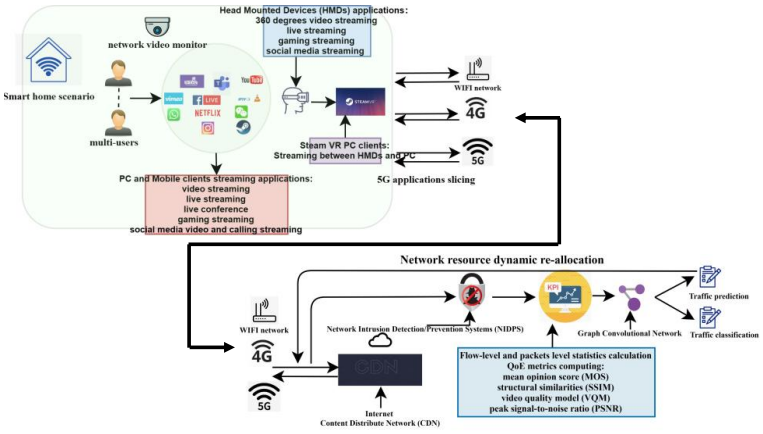


Fig. 1. Flow chart of traffic classification and prediction for streaming applications in heterogeneous network

[Publication 6]: Encapsulated and Anonymized Network Video Traffic Classification With Generative Models

Chen, T., Grabs, E., Pētersons, E., Efrosinin, D., Ipatovs, A., Klūga, J. Encapsulated and Anonymized Network Video Traffic Classification with Generative Models. In: 2022 Workshop on Microwave Theory and Techniques in Wireless Communications 2022 (MTTW'22): Proceedings, Latvia, Riga, 5-7 October, 2022.

Encapsulated and anonymized network video traffic classification with generative models

Tianhua Chen
Telecommunications Institute
Riga Technical University
Riga, Latvia
tianhua.chen@edu.rtu.lv

Elans Grabs
Telecommunications Institute
Riga Technical University
Riga, Latvia
elans.grabs@rtu.lv

Ernests Petersons
Telecommunications Institute
Riga Technical University
Riga, Latvia
ernests.petersons@rtu.lv

Dmitry Efrosinin
Institute Institute of Stochastics
Johannes Kepler University
Linz, Austria
dmitry.efrosinin@jku.at

Aleksandrs Ipatovs
Telecommunications Institute
Riga Technical University
Riga, Latvia
aleksandrs.ipatovs@rtu.lv

Janis Kluga
Telecommunications Institute
Riga Technical University
Riga, Latvia
janis.kluga@rtu.lv

2022 Workshop on Microwave Theory and Techniques in Wireless Communications (MTTW) | 978-1-6654-6439-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/MTTW56973.2022.9942564

***Abstract*—In recent years, anonymization and encrypted web applications have become increasingly popular among the public. Although the Internet communication process already has complete privacy protection, network security technologies continue to evolve, and technologies for exploring the commercial behavioral characteristics of network users' fingerprints are also accelerating. In this paper, we picked benchmark public datasets, including anonymously viewing video streaming traffic using The Onion Router (Tor) network technology, establishing tunnel encapsulated video streaming packets using Virtual Private Network (VPN) technology, and viewing video streaming traffic scenarios in normal mode. We combined video streaming application types and names to construct a new dataset for multi-class classification problem. Unlike the frequently used classification discrimination models, we used generative models, including Restricted Boltzmann Machines (RBM) and Deep Belief Networks (DBN), having an advantage of features extraction learning to construct models for probabilistic judgments, and better performance, that has been verified and supported in several works. The experimental results show the classification accuracy is as high as 0.94 for eight categories and 0.97 for three categories. Its overall classification performance is good, and different classification performance metrics are balanced.**

Keywords*—*Traffic Classification, Restricted Boltzmann Machines, Deep Belief Networks, Tor network

I. Introduction

Streaming traffic has been an essential category in studying Internet traffic classification, and it includes video versus audio traffic data which will be presented with label of the same or different type for subsequent classification. The video streaming category usually demands the maximum network resources and lasts for a considerable period, researchers detect and strip out video streaming applications or categories through a comprehensive analysis of such transmission process characteristics encompassing upstream and downstream throughput, packet transmission intervals, and some subjective and objective metrics – Quality of Service (QoS) and Quality of Experience (QoE). However, the demand for encryption and privacy protection of network traffic data makes people more cautious when watching streaming videos. Although the current communication process between clients and streaming servers utilizes encrypted network protocols Transport Layer Security/Secure Sockets Layer (TLS/SSL), the payload of video traffic packets cannot be read. However, websites can still identify users by browser fingerprinting or using Deep Packet Inspection (DPI) tools to identify some signatures and packet headers to determine users' behavior [1].

Common ways to prevent fingerprint generation include installing browser plug-in tools to disable trackers from running on websites. Browsers also allow users to browse in incognito mode, and most users will use the default configuration information to make the generated fingerprints more similar. Alternatively, browsers do not allow websites to use JavaScript, which carries many fingerprint information. In addition, establishing a secure tunnel using a Virtual Private Network (VPN) can hide the actual Internet Protocol (IP) address, reduce individual information in browser fingerprint profiles, and prevent Internet Service Providers (ISPs) censorship or targeted advertising. The Onion Router (Tor) network can be used to build anonymous communication to avoid network track; similar to the browser's incognito mode trait, the fingerprint generated using the Tor browser's default configuration is the same as all other Tor browsers reducing the uniqueness of the fingerprint [2]. Various methods and tools can enhance user privacy and minimize the likelihood of being identified, and there are pros and cons between them all. From the network traffic analysis perspective, Tor and VPN are the two methods of most concern, and how to identify them becomes a difficult point.

Using VPN technology to watch streaming video requires third-party software to establish proprietary tunnels in advance. Although the video traffic data transmitted by the tunnel cannot be monitored directly, encrypted end-to-end communication will still generate browser fingerprints and leak users' privacy information. However, VPN software developers will continue improving VPN security, offering many advanced features, such as ad-blocking, forward secrecy, more robust private VPN protocols, and Tor over VPN. VPN client, server IP addresses, and port configured with the help of software, complete communication process information will be mastered by the software providers.

Tor is composed of thousands of relay nodes and bridge servers. It would not establish direct communication between the browser client and server and randomly send request information to the first guard relay node to set up a TLS connection while watching streaming videos. After that, traffic data will be transferred to the second middle relay node

This work has been supported by the European Regional Development Fund within the Activity 1.1.1.2 "Post-doctoral Research Aid" of the Specific Aid Objective 1.1.1 "To increase the research and innovative capacity of scientific institutions of Latvia and the ability to attract external financing, investing in human resources and infrastructure" of the Operational Programme "Growth and Employment" (No.1.1.1.2/VIAA/2/18/332)

978-1-6654-6439-0/22/$31.00 ©2022 IEEE

and third exit node step by step, and the exit node then transfers data information to the streaming server. The most significant advantage is that three different TLS connections are established during three relay node forwards, thus isolating communication contents leakage between skipped nodes, while client information is also anonymized three times during different relay node. Both requests for video contents and server responses are encrypted and anonymized multiple times, declining ISP monitoring scrutiny and production of unique client browser fingerprints. However, the attendant drawback is that network bandwidth resources for relay node servers are severely limited, and customers can not view some streaming videos with high-definition resolutions and frame rates.

In the context of the development of encryption encapsulation and anonymization techniques, it has become more challenging to classify video streaming category traffic under different scenarios. This paper uses generative machine learning models to classify video streaming category traffic encapsulated by OpenVPN, video streaming category traffic captured in Tor networks, and normal/standard mode (Non-VPN and Non-Tor) captured video streaming traffic. Furthermore, classification performance metrics are explored in conjunction with different applications.

II. Related Work

Currently, most of the existing work focuses on identifying Tor traffic or VPN traffic from mixed Internet traffic and classifying it according to relevant features, such as applications or traffic categories. Bai et al. [3] matched several Tor traffic fingerprints, including characteristic strings, packet bytes, and the transmitted packets period, to identify whether it is Tor traffic. They got a 95.98% recognition rate. Similarly, Mayank et al. [4] identified Tor traffic using the TLS connection process, including ClientHello, ServerHello, Certificate message, and other key field features for matching. Shahbar et al. [5] proposed circuit level and flow level classification methods, that achieved good classification metrics in Browsing, Streaming, and BitTorrent, i.e. Tor traffic dataset consisted of three categories. In which a single circuit consists of a client and three relay nodes, and their features cover relay cells per circuit lifetime, the rate of the downlink cells to the uplink cells, etc. Using circuit-level and flow-level features, 100% classification accuracy was achieved under C4.5 and Bayes Net classifiers, respectively. Jia et al. [6] extracted packet lengths and packet forwarding delay related statistical features and achieved Tor traffic recognition rate of up to 99% under the improved decision tree algorithm and 94% classification accuracy in four Tor traffic applications scenario. Juan et al. [7] also exploited flow level time-dependent statistical features for Tor-Meek traffic identification and achieved over 94% precision and recall scores when using C4.5, Random Forest, and K-Nearest Neighbors (KNN) classifiers.

Islam et al. [8] creatively classified VoIP category traffic in mixed VPN traffic, Tor traffic, and standard traffic datasets, achieving over 96% classification accuracy with a One Dimensional Convolutional Neural Network (1D-CNN) classifier and 60 flow-level features. Ghanavi et al. [9] improved and classified 22 applications in the mixed VPN and Tor traffic datasets, which obtained over 93% classification accuracy combined with 266 flow-level features and Generative Adversarial Classification Network (GACN). Shapira et al. [10] performed a breakdown of VPN and Tor traffic applications and categories in which overall 83% classification accuracy in the Tor, VPN, and Non-VPN composed mixed traffic dataset. Besides, they also achieved over 99.7% classification accuracy for ten applications and categories in the mixed traffic dataset. They used payload size distribution, packet transmission intervals, and other size-related and time-related features in the flow level and linked LeNet-5 CNN classifiers. Most research works have achieved better classification metrics, but the shortcomings were limited to the detection and classification of traffic under one of the specific scenarios, while the granularity of the classification is also coarser. We constrain the same streaming traffic category variable and correspond to traffic scenario features to applications one-to-one, using generative models combined with prior probabilities for easier portraying of the similarity of the same traffic data category, thus effectively improving the classification metrics.

III. Methodology

A. Datasets

To achieve the goal of classifying mixed streaming traffic under multiple scenarios, we considered using public benchmark datasets, including Canadian Institute for Cybersecurity (CIC) gathered the ISCX VPN-nonVPN traffic dataset [17] and the ISCX Tor-nonTor dataset [18]. These two datasets all covered video streaming category traffic in Tor, VPN, and standard mode scenarios. We picked Vimeo, YouTube, and Netflix three applications traffic under the standard and VPN scenarios, respectively. Similarly, using Vimeo and YouTube applications traffic under the Tor scenario, but Netflix application traffic samples are not included in the Tor-nonTor dataset, so a total of 8 label categories. On this basis, we will narrow the scope of the label, delete the application name information, and directly classify the video traffic named after the three major scenarios.

Fig. 1 shows the complete flow chart of mixed video streaming traffic scenarios classification. We create a mixed traffic dataset containing eight video streaming traffic categories according to the above introduced characteristics of benchmark datasets. Secondly, feature selection will be performed to reduce the number of features and dimensionality, making the training model more generalizable and reducing overfitting. The following step focuses on data preprocessing which can improve computational efficiency and optimize training results, which include data resampling and normalization steps. After data preprocessing, the new dataset will be randomly split into a training dataset and a testing dataset, and their sample size allocation ratios are 70% and 30%, respectively. The next stage will train the training dataset in configured generative models and obtain prediction results, and it will be evaluated combined with the testing dataset in multiple metrics.

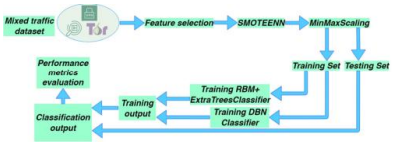


Fig. 1. Flow chart of mixed video streaming traffic scenarios classification

B. Generative models

The most representative generative models include Restricted Boltzmann Machines (RBM) and Deep Belief Networks (DBN). RBM [11] is a probabilistic graph model interpreted in terms of random neural networks, consisting of an observation layer and a hidden layer, with no connections within layers and full connections between two hidden and observation layers. The graph corresponding to RBM is a bipartite undirected graph model, and its advantage is that the utilization of hidden variables can describe complex data distributions, learn the internal feature representation of data, and handle missing or irregular data. However, a single RBM as a feature extractor cannot satisfy the mission of classification output, and an extra tree classifier is raised as a classification discriminant in this paper. The DBN [12] can be formed by stacking multiple RBMs, and adding multiple hidden layers can obtain higher performance gains. DBN training process can be banded into two stages: layer-by-layer pre-training and fine-tuning. The model parameters are initialized to better values by greedy pre-training, and then the parameters are fine-tuned by the backpropagation algorithm combined with the softmax function to solve the output category in the supervised learning tasks [13]-[14].

According to the characteristics of the above models, we developed two classifiers, including a single Bernoulli RBM plus ExtraTreesClassifier [15] and DBN Classifier [16] with specific parameters and configurations information as shown in Table I. For the single Bernoulli RBM model, the number of binary hidden layer units is closely related to the calculation of the joint probability distribution of the observation and hidden layers, and the abstraction of the input visual layer data can find more complex and profound dependencies. The number of hidden layers units is determined to be 300 after several iterations of calculation. RBM finds the optimal parameters, including weights and biases by maximizing the likelihood function and increasing the number of updates that iterations facilitate, allowing Gibbs sampling to generate better samples that match the joint probability distribution. We set a 0.02 low learning rate for weight update combined with ten iterations over the training dataset to help effectively extract visible layer input features. The training output connects an ensembled extra trees classifier to calculate the Gini impurity metric for classification output. DBN Classifier comprises two hidden layers with the same default 256 hidden units, similar to RBM with a 0.02 low learning rate and 30 iterations for pre-training. We use the ReLU function as the activation function of the hidden layer unit, which has the advantage that the sparse model can better mine relevant features and does not suffer from gradient saturation and disappearance. The output will be supervised fine-tuning combined with 100 backpropagation iterations and will use the Softmax activation function.

TABLE I: CLASSIFIER A AND B PARAMETERS SETTING.

ClassifierA	Parameters	ClassifierB	Parameters
Bernoulli RBM + ExtraTrees Classifier	n_components: 300	DBN Classifier	hidden_layers_structure: [256, 256]
	learning_rate: 0.02		learning_rate_rbm: 0.02
	n_iter: 10		learning_rate=0.1
	n_estimators: 180		n_epochs_rbm=30
	criterion: 'gini'		n_iter_backprop: 100
	random_state: 0		batch_size: 32
	verbose: 1		activation_function: 'relu'
			dropout_p: 0.2

C. Feature selection

Flow is the basic granular level of network traffic which contains two communication ends transmitted packets, researchers can perform traffic monitoring and analysis on this basis. CIC developed a corresponded traffic flow generator CICFlowMeter for collected datasets and provided nearly 80 flow-level traffic features for further analysis by the researchers. However, another flow generator NFStream [19] can offer close to 90 flow-level traffic features, and it has an integrated nDPI tool for deep packet inspection, dramatically improving the identification and verification of video streaming traffic. Those features can be summarized as flow-related core features, statistical analysis features, and other ground-truth features. In the actual model training process, only numerical features are selected. Therefore, selected features are divided into three significant directions according to the direction of packet flowing: source to destination, destination to source, and bidirectional, combining their relevant statistical bytes, durations, and packet characteristics, a full of 55 features. One of the most significant attributes of video streaming traffic is that a single flow exporting will continues for an extended period because most packets have duplicate flow tuples lead flow into the same flow pipeline. We set flow metering expired period is one second which effectively shortens the flow generation period to avoid too much data in a single flow and acquires more flow data samples.

Our previous works [20]-[21] used packet-based traffic intensity features to achieve good classification metrics, which will be chosen for comparison with the flow-level features in this paper. The primary feature selection step includes calculating each video streaming traffic category intensity with different sampling time intervals and generating an integrated intensity matrix of all other categories for training. Table II. shows detailed elected features of both packet-level and flow-level.

D. Data preprocessing

Limited by the number of samples in the public benchmark datasets, the distribution of the total number of samples in each category varies significantly in the mixed traffic dataset. Dataset samples will be over-sampling using Synthetic Minority Oversampling TEchnique (SMOTE) and cleaning using Edited Nearest Neighbors (ENN) [22]. Oversampling means increasing the number of samples of all minority classes except the majority class. SMOTE as a representative oversampling algorithm, calculates the five nearest neighbors of each minority class sample, randomly selects N samples from the five nearest neighbors for random linear interpolation, and finally adds the simulated new samples to the dataset. After that, calculate three nearest neighbor samples of the majority class whether different from their

TABLE II: TWO LEVELS OF TRAINING FEATURES.

Level	Features
Flow	Src_port, dst_port, ip_version, bidirectional, src2dst and dst2src bytes, flow durations and packets cumulative count
	Min, mean, stdev and maximum of bidirectional, src2dst and dst2src bytes and flow durations cumulative count
	TCP SYN, CWR, ECE, URG, ACK, PSH, RST and FIN flag set packets cumulative count of bidirectional, src2dst and dst2src
Packet	Packets traffic intensity vectors composed feature matrix

categories. We will remove the majority class of all samples to achieve a balanced sample size if they are not the same [23]. The combination of oversampling and undersampling methods makes the categories in the original data no longer seriously imbalanced and prevents the data from overfitting.

The next step is to normalize the dataset, which can not only improve the convergence speed of the model but also prevent the gradient explosion of the DBN model. The normalized scaling interval will be selected as [0-1], which is also determined by the RBM model's binary input in visible and hidden units. Equation 1 shows the calculation process, which takes all numerical features in each row sample, selects the maximum and minimum values, and substitutes the value of each feature into the equation to find the normalized new value.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

where x is the value of a single feature for each sample, x_{max} is the maximum value of all features for each sample, x_{min} is the minimum value of all features for each sample and x_{scaled} is the normalized value of a single feature for each sample.

IV. Experiment Result Analysis

When using the packet-level direct traffic intensity method as a feature for classification, evaluating whether the hyperparameter traffic intensity sampling time intervals (Tsmp) strongly impact the classification results is also under investigation. Therefore, based on all relevant configuration information, we will perform a total of 20 classification training sessions. In addition, classification evaluation metrics selection is also crucial to investigate the classification performance of the hybrid video streaming traffic, Grandini et al. [24] gave an overview survey of metrics used in the multi-class classification task, and the classification accuracy will be studied first. Other metrics include weighted recall and precision scores, Macro F1-score and Micro F1-score. We also choose a 10-Folds cross-validator to split the training dataset into training and testing subsets. Each fold is considered a testing subset; other folds are training subsets repeated ten times. Therefore will obtain ten times cross-validated classification accuracy and their statistics mean and standard deviation values. Constrained by the performance of the DBN classifier, we can not perform a 10-Folds cross-validation calculation and achieve relevant metrics.

Table III. presents both classifiers all twenty times classification training output metrics summarization, BernoulliRBM plus ExtraTreesClassifier composed classifier A achieve the best classification accuracy 0.94 and 0.00 standard deviation of 10-Folds cross-validation accuracy under flow-based method among all eight classes. Similarly, we obtained 0.97 classification accuracy and 0.00 standard deviation of 10-Folds cross-validation accuracy in all three classes' classification tasks under the same classifiers and features. Considering each category's classification accuracy, they all achieved as high as the same precision, recall, and F1-score value with classification accuracy in two kinds of categories under the same classifiers and features, indicating the effective classification.

For the direct traffic method, from the parameter Tsmp decreasing, the overall classification accuracy appears to increase trend in two kinds of categories which due to the formula defined in [21], the smaller the value of Tsmp as the denominator, the more features will be used, with a maximum of more than 10,000 features trained. Nevertheless, there is still instability with the 10-Folds cross-validation accuracy mean, and standard deviation values differ significantly from classification accuracy. Besides, each class classification accuracy imbalance problem exists, as the table shows a high classification accuracy of 0.91 for the eight classes but only gets a 0.74 macro f1-score which demonstrates the gap between different classes to obtain f1-scores is large, and this is a significant disadvantage compared to the flow-based method.

The highest acquired classification accuracy is 0.91 and 0.85 macro f1-score in 8 classes task under the direct traffic intensity method and DBN classifier, which has a slight disparity compared to classifier A. We also obtained as high as 0.92 classification accuracy in 3 classes under the flow-based method and DBN classifier. Overall, good classification

TABLE III: Classifier A and B classification metrics summarization.

Classifiers	Features	Category	Accuracy	Precision	Recall	F1-Macro	F1-Micro	Twnd	Tsmp	10Fold-Mean
BernoulliRBM+ExtraTreesClassifier	Direct traffic intensity method	8 classes	0.75±0.07	0.75	0.75	0.74	0.75	10s	1000ms	0.70
			0.87±0.02	0.88	0.87	0.82	0.87	10s	100ms	0.84
			0.92±0.02	0.92	0.92	0.89	0.92	10s	10ms	0.91
			0.91±0.02	0.91	0.92	0.74	0.92	10s	1ms	0.93
		3 classes	0.82±0.01	0.81	0.82	0.70	0.82	10s	1000ms	0.79
			0.84±0.01	0.84	0.84	0.82	0.84	10s	100ms	0.82
			0.88±0.03	0.88	0.88	0.76	0.88	10s	10ms	0.91
			0.90±0.03	0.90	0.90	0.69	0.90	10s	1ms	0.88
	Flow based method	8 classes	0.94±0.00	0.94	0.94	0.94	0.94			0.95
		3 classes	0.97±0.00	0.97	0.97	0.97	0.97			0.97
DBN Classifier	Direct traffic intensity method	8 classes	0.68	0.70	0.69	0.66	0.69	10s	1000ms	
			0.66	0.70	0.66	0.60	0.66	10s	100ms	
			0.91	0.93	0.91	0.85	0.91	10s	10ms	
			0.90	0.90	0.90	0.58	0.90	10s	1ms	
		3 classes	0.79	0.79	0.79	0.72	0.79	10s	1000ms	
			0.70	0.76	0.70	0.66	0.70	10s	100ms	
			0.86	0.90	0.86	0.83	0.86	10s	10ms	
			0.89	0.90	0.89	0.85	0.89	10s	1ms	
	Flow based method	8 classes	0.75	0.77	0.75	0.75	0.75			
		3 classes	0.92	0.92	0.92	0.91	0.92			

metrics are achieved using both of our configured generative model classifiers. For the selected training features, the flow-based approach achieves better classification accuracy and stable performance, while using the packet-based approach also achieves higher classification accuracy, but with large standard deviations for multiple cross-validations and unstable classification performance.

For the composite video streaming traffic, we can learn each class classification metric from the normalized confusion matrix of classifier A under the flow-based method, as shown in Fig. 2. Except for the Tor_Vimeo class and classification accuracy of other classes exceeds 90%. The main reason is attributed to the uneven distribution and the small number of samples in the original dataset. After completing the data preprocessing, the new instances have a high similarity of features leading to difficulties in classification. Fig. 3 shows the normalized confusion matrix of classifier A under the flow-based method with three classes task. All classes achieved over 95% classification accuracy with more minor indicator differences between categories. The imbalance between the sample data volumes of different categories is more serious on classification performance, and one of our future work directions will reconstruct samples of the dataset to capture and use more video streaming traffic data to complement the public datasets and improve classification performance metrics.

In general, we can learn that the poor performance of DBN classification is limited by the training dataset size since the weights obtained in pre-training are not close to the optimal weights and thus affect the model's performance. The advantage of ExtraTrees is that it uses a random feature and a random threshold to classify the decision tree classifier nodes. The variability between different decision tree classifiers and the randomness becomes superior. It provides additional randomness, suppresses overfitting, and reduces variance.

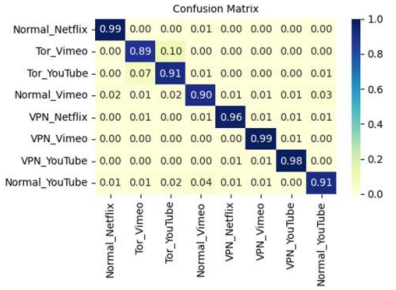


Fig. 2. Normalized confusion matrix of classifier A under the flow-based method with eight classes

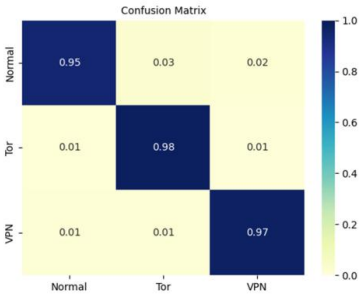


Fig. 3. Normalized confusion matrix of classifier A under the flow-based method with three classes

V. Conclusion

This paper elaborates on using generative models to classify mixed video streaming media traffic datasets and achieves classification accuracy of over 90%. VPN, Tor, and normal mode three scenarios streaming media traffic all reap good overall classification performance and efficiency. We chose RBM and DBN as representative generative models, and the advantage lies in feature extraction. The more the number of features used for training it will get better the classification performance. However, using the single packet-level features method will cause high feature similarity between each column, and the threshold range of the Tsmp parameter needs to be reasonably tuned, which helps determine the appropriate number of features to input. In addition, when using the flow-based method can achieve high classification indicators in the two classifiers. Still, due to the imbalance in the number of training samples and distribution, the error loss in the backpropagation process in the DBN classifier is significant, resulting in inaccurate classification output. It is necessary to increase the number of iterations and the number of samples for further validation. The current work uses offline public datasets, and future work will continue to explore the possibility of real-time classification and performance metrics.

References

[1] R. T. El-Maghraby, N. M. Abd Elazim, and A. M. Bahaa-Eldin, "A survey on deep packet inspection," in *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, Dec. 2017, pp. 188–197. doi: 10.1109/ICCES.2017.8275301.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," San Diego, CA, Aug. 2004. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/Tor-second-generation-onion-router

[3] X. Bai, Y. Zhang, and X. Niu, "Traffic Identification of Tor and Web-Mix," in *2008 Eighth International Conference on Intelligent Systems Design and Applications*, Kaohsuing, Taiwan, Nov. 2008, pp. 548–551. doi: 10.1109/ISDA.2008.209.

[4] P. Mayank and A. K. Singh, "Tor traffic identification," in *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*, Nagpur, Nov. 2017, pp. 85–91. doi: 10.1109/CSNT.2017.8418516.

[5] K. Shahbar and A. N. Zincir-Heywood, "Benchmarking two techniques for Tor classification: Flow level and circuit level classification," in *2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, Orlando, FL, USA, Dec. 2014, pp. 1–8. doi: 10.1109/CICYBS.2014.7013368.

[6] J. Lingyu, L. Yang, W. Bailing, L. Hongri, and X. Guodong, "A hierarchical classification approach for Tor anonymous traffic," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, Guangzhou, May 2017, pp. 239–243. doi: 10.1109/ICCSN.2017.8230113.

[7] W. Juan, C. Shimin, Z. Jun, H. Bin, and S. Lei, "Identification of Tor Anonymous Network Traffic Based on Machine Learning," in *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, Dec. 2021, pp. 150–153. doi: 10.1109/ICCWAMTIP53232.2021.9674056.

[8] F. U. Islam, G. Liu, J. Zhai, and W. Liu, "VoIP Traffic Detection in Tunneled and Anonymous Networks Using Deep Learning," *IEEE Access*, vol. 9, pp. 59783–59799, 2021, doi: 10.1109/ACCESS.2021.3073967.

[9] R. Ghanavi, B. Liang, and A. Tizghadam, "Generative Adversarial Classification Network with Application to Network Traffic Classification," in *2021 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, Dec. 2021, pp. 1–6. doi: 10.1109/GLOBECOM46510.2021.9685899.

[10] T. Shapira and Y. Shavitt, "FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification," *IEEE Trans. Netw. Serv. Manage.*, vol. 18, no. 2, pp. 1218–1232, Jun. 2021, doi: 10.1109/TNSM.2021.3071441.

[11] A. Fischer and C. Igel, "Training restricted Boltzmann machines: An introduction," *Pattern Recognition*, vol. 47, no. 1, pp. 25–39, 2014, doi: https://doi.org/10.1016/j.patcog.2013.05.025.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[13] Yuming Hua, Junhai Guo, and Hua Zhao, "Deep Belief Networks and deep learning," in *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, Harbin, China, Jan. 2015, pp. 1–4. doi: 10.1109/ICAIOT.2015.7111524.

[14] R. Malik, Y. Singh, Z. A. Sheikh, P. Anand, P. K. Singh, and T. C. Workneh, "An Improved Deep Belief Network IDS on IoT-Based Network for Traffic Systems," *Journal of Advanced Transportation*, vol. 2022, pp. 1–17, Apr. 2022, doi: 10.1155/2022/7892130.

[15] L. Buitinck *et al.*, "API design for machine learning software: experiences from the scikit-learn project," *arXiv preprint arXiv:1309.0238*, 2013.

[16] Albertbup, "A Python implementation of Deep Belief Networks built upon NumPy and TensorFlow with scikit-learn compatibility." 2017. [Online]. Available: https://github.com/albertbup/deep-belief-network

[17] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[18] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, A. A. Ghorbani, and others, "Characterization of Tor traffic using time based features.," in *ICISSp*, 2017, pp. 253–262.

[19] Z. Aouini and A. Pekar, "NFStream A flexible network data analysis framework," *Computer Networks*, vol. 204, p. 108719, Feb. 2022, doi: 10.1016/j.comnet.2021.108719.

[20] E. Grabs, E. Petersons, A. Ipatovs, and D. Chulkovs, "Supervised Machine Learning based Classification of Video Traffic Types," in *2020 24th International Conference Electronics*, Palanga, Lithuania, Jun. 2020, pp. 1–4. doi: 10.1109/IEEECONF49502.2020.9141625.

[21] E. Grabs *et al.*, "Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks," in *2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW)*, Riga, Latvia, Oct. 2021, pp. 58–63. doi: 10.1109/MTTW53539.2021.9607153.

[22] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20–29, Jun. 2004, doi: 10.1145/1007730.1007735.

[23] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972, doi: 10.1109/TSMC.1972.4309137.

[24] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," 2020, doi: 10.48550/ARXIV.2008.05756.

[Publication 7]: Multiclass Live Streaming Video Quality Classification Based on Convolutional Neural Networks

Chen, T., Grabs, E., Pētersons, E., Efrosinin, D., Ipatovs, A., Bogdanovs, N., Rjazanovs, D. Multiclass Live Streaming Video Quality Classification Based on Convolutional Neural Networks. Automatic Control and Computer Sciences, Vol. 54, No. 5, pp.455-466. ISSN 0146-4116. e-ISSN 1558-108X, 2022.

ISSN 0146-4116, Automatic Control and Computer Sciences, 2022, Vol. 56, No. 5, pp. 455–466. © Allerton Press, Inc., 2022.

Multiclass Live Streaming Video Quality Classification Based on Convolutional Neural Networks

T. Chen[a, *], E. Grabs[a], E. Petersons[a], D. Efrosinin[b], A. Ipatovs[a], N. Bogdanovs[a], and D. Rjazanovs[a]

[a] *Telecommunications Institute, Riga Technical University, Riga, LV-1048 Latvia*

[b] *Institute of Stochastics, Johannes Kepler University, Linz, A-4040 Austria*

**e-mail: Tianhua.Chen@edu.rtu.lv*

Received November 1, 2021; revised December 16, 2021; accepted December 17, 2021

Abstract—E-sports live streaming video is rapidly coming into people's lives. High-quality video is an essential factor affecting users' perception. This paper presents conventional network traffic analysis methods for traffic intensity selection as a feature combined with deep learning classifiers for streaming videos classification with different resolutions and frame rates per second. According to the experimental results, the convolution neural networks showed the best results in multiclass classification with accuracy as high as 97%. This superiority can help E-sports operators to improve the quality of live streaming videos and provide differentiated services for their users. Furthermore, the article describes research on the performance of various deep learning classifiers with different hyperparameters. The number of filters in convolution layers and training batch size can significantly affect classification performance according to testing results. It is still necessary to avoid hyperparameters' designated values significantly influencing the classification results.

Keywords: multiclass classification, traffic intensity, convolutional neural networks, deep learning, features selection, hyperparameters evaluation

DOI: 10.3103/S0146411622050029

1. INTRODUCTION

E-sports have been experiencing accelerated streaming media expansion. More and more people have become used to watch E-sports live streaming on different platforms, including YouTube, Twitch, etc. League of Legends is one of the most popular E-sports games and there are 12 professional leagues for League of Legends worldwide; so, people can watch their loved qualification trials on live streaming platforms using content delivery network (CDN) technique. However, different leagues are holding by different operators, and there are no uniform standards for E-sports live streaming video quality. This paper proposes a novel method for the classification of E-sports video streams with different quality modes.

A review of existing literature studies shows that video classification is not a brand-new topic, and three methods can briefly summarize it. Traditional research on graph processing is an essential direction for computer vision. When using graph content as features for machine learning and classification, segmented video frames can implement this kind of function similarly to video content. Authors in [1–3] investigated image and video datasets with deep learning architectures by selecting video spatio-temporal volumes and patterns as features and using natural scene statistics properties, action recognition, motion with rank pooling, and acoustic clues popularly. The convolutional neural network has significant advantages in image feature extraction. 2D-CNN focuses on extracting frame graph contents, and 3D-CNN adds one more dimension for motion or spatial information. CNN captured features combined with LSTM, GRU, or SVM classifiers for training and classification. Authors in [4] extracted semantic information as features and captured audio-visual content, as well as auxiliary sensor data as the second feature. However, it was still depended on spatio-temporal visual and motion features. Authors in [5, 6] selected features according to camera motion, subject motion, speed, direction, acceleration, averaged luminance and saturation, audio, and the number of flashlights. Moreover, it needs to be combined with specific video genres. Authors in [7, 8] utilized pattern categorization techniques, more specifically, consisting of the strength of quality fluctuation combined with a categorization of the encountered fluctuation pattern, or video-coded information extraction at macroblock (MB) level with no-reference objective peak signal to noise ratio (PSNR) as features.

The second method of video classification is based on network traffic analysis. The potential features are built upon different video streaming techniques according to traffic patterns [9], characteristics of video flow changes during the transmission process, statistic characteristics of different OSI layers or protocol data. The difference of the traffic applications and models can also be considered in [10–14] or feature subsets fusion method for reducing the number of dimensions [15, 16]. Authors in [17, 18] combined specific streaming technique characteristics with traffic flows information. In general, if considered from the network traffic perspective, multiple different traffic characteristics can be selected as video classification features. However, sometimes it causes feature explosion problems. Therefore, [19] proposed an alternative method that converts traffic packet characters, including source and destination host destination, port number, packet header, and payload information into gray-scale graphics and uses CNN extract features for classification. [20] use nDPI tool that identifies traffic flow straightforward and automatically assigns labels with corresponding protocols, TCP flow data make up single dimension dataset which, using 1D-CNN, can finish classification task.

Video classification based on quality of service (QoS) and quality of experience (QoE) has become a prominent hot spot; it is related to video traffic QoS guarantee and focuses on video content and quality evaluation. For this type of features selection, [21] proposed to split downstream and upstream rates into different QoS flow aggregation categories, [22] utilized downstream statistical characteristics and upstream statistical characteristics associated with local IP downloads and divided them into different categories of video flows. Authors in [23] developed a deep learning framework for QoE prediction, including mean opinion scores, resolution, frame per second, coding compression rates, and bitrates used for features processing; [24, 25] focused on video source coding, packet classification, error concealment within the cost distortion optimization framework, and set different penalty factors with video diagnosis distortion. Authors in [26] exploited PSNR scores standard combined with frame rate, sending bitrate, packet error rate, and bandwidth. After that, the selected video clips were grouped into four clusters according to content type. Synthesize, the method based on QoS and QoE can achieve more possibilities for features selection; it combines video quality evaluation methodology and traffic analysis.

To sum up the previously introduced features selection methods for video classification, direct video traffic flows statistical data utilization will be the most convenient and efficient method; the research of the presented article will mainly focus on selecting traffic intensity as a feature for supervised learning. Most video classification models were based on binary or 3-class classification, and their feature difference usually was apparent. Therefore, it will be more complicated when the problem evolves to the higher classes' classification. According to E-sports live streaming video network traffic difference, we split into four different quality classes and obtained the highest classification accuracy, close to 98%. For the multiclass classifier models, including deep neural networks (DNN), convolution neural networks (CNN) architectures have been considered.

2. CONVOLUTIONAL NEURAL NETWORKS

DNN uses new activation functions and optimizers and has excellent feature learning ability. By constructing machine learning models with multiple hidden layer nodes and massive training data, as well as layer by layer feature transformation, the feature representation of the sample in the original space is transformed into a new feature space to learn more valuable features, which can effectively overcome the problem of gradient disappearance. This can improve the accuracy of classification or prediction. However, DNN has the problem of parameter expansion, and it is easy to get overfitting and fall into optimal local solution after training. CNN can overcome these problems. The typical structure of CNN consists of convolution layer, nonlinear layer, pooling layer, and full-connected layer.

The convolution layer can extract local features without fixing the size of feature map input, because it slides the window of local areas and can share weights. The full-connected layer obtains the global information, its weight is nonadjustable, and the size of the input feature map must also be determined. Each node of the full-connected layer is catenated with all nodes of the previous layer, so the convolution layer is more efficient in some cases. The activation function of the convolution layer uses a rectified linear unit function (ReLU), which is similar to the traditional activation function, such as sigmoid or tanh functions, but linear rectification functions can more efficiently calculate gradient descent and backpropagation, avoid gradient explosion and gradient disappearance, and simplify the calculation process. The pooling layer reduces the dimension of feature space on the convolution neural network but will not reduce the depth. When the maximum pooling layer is applied, the maximum number of input regions is restored; when average pooling is utilized, the average value of the input area is exploited. In general, the convolution layer of the convolution neural network has significant advantages in feature extraction. The existing literature has shown that the extracted features can be combined with various classifiers and applied in various scenes with more robust adaptability.

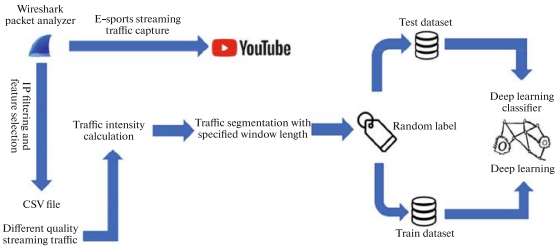


Fig. 1. The flowchart of experiment and data preparation.

3. EXPERIMENTAL SETUP

The primary purpose of the presented research is to categorize different quality live streaming E-sports live streams. The evaluation indicators for the quality of live video include encoding rate, resolution, and frame rate. Common coding rates include variable bitrate, average bitrate, and constant bitrate. The more data bits transmittal per unit of time, the smaller the compression ratio and video distortion will be. Video resolution is a crucial parameter to measure the volume of data for streaming images. Common video resolution formats include 720p, 1080p, and 1440p. Currently, YouTube supports up to 8K ultra high definition (UHD, 4320p) video resolution. However, live video is still mainly being streamed in HD and Full HD modes. The video frame rate refers to the number of frames displayed per second (FPS). A high frame rate can offer smoother and more realistic graphics, essential for users who watch live streaming E-sports videos. The current YouTube videos already support up to 60FPS, and most operators also provide such options.

For E-sports live streaming video to be classified, the three indicators mentioned above must be considered. Since the video coding format is H.264 in most cases, the choice of coding rate is uncertain. For video resolution, the values 720p, 1080p were chosen, whereas values for video frame rate are 30FPS, 60FPS. All video fragments originate from the YouTube platform, and their content is related to League of Legends contests. All videos used for experiments were released by different league operators and were randomly selected.

The video classification is based on network traffic analysis, and Wireshark was be used to capture live streaming E-sports video traffic. The flowchart of the performed experiment is displayed in Fig. 1; the classes of traffic are represented by different video quality, which includes 720p@30FPS, 720p@60FPS, 1080p@30FPS, and 1080p@60FPS. The first step was to capture video traffic for all classes and do relevant traffic IP filtering.

The next step is sampling captured traffic over 10ms time intervals and generating time series as several packets per sampling period (i.e., traffic intensity). This traffic intensity vector is exported in CSV format for each category of videos. The length of the sliding time window is also an essential parameter, as it defines the number of features and reshapes the traffic intensity vector into a matrix of features. The rows of this matrix of features contain traffic intensity for random video classes, and the labels vector is accordingly formed. The final step is using this dataset for deep learning classification models training and testing.

4. METHODOLOGY

Video traffic intensity was chosen to be a classification feature, and it is assessed by counting the number of packets per sampling time interval. According to the wireshark input and output graphs, the sampling time intervals (T_{SMP}) can be in the range of 1ms to 10 min. The custom-made program in Python allows selection of any value for this parameter, given the fact there is enough captured traffic packets. In the present article, the value of 10 ms has been selected. The traffic intensity vector $\mathbf{x}$ can be represented as a column of intensity values, as is shown by formula (1):

$$\mathbf{x} = (x_1, x_2, \cdots, x_m)^{\mathrm{T}}, \tag{1}$$

where m represents the total number of 10 ms sampling time intervals, and x_m is the number of packets captured during mth sampling time interval. Such traffic intensity vector needs to be created for each class of captured traffic data. Let us denote by $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ the corresponding intensity vectors calculated for four video quality net-

work traffic, respectively: 720p@30FPS, 720p@60FPS, 1080p@30FPS, and 1080p@60FPS. Similarly, *j, k, l* and *n* will also stand for the corresponding number of 10 ms sampling time intervals.

$$\mathbf{a} = (a_1, a_2, \ldots, a_j)^{\mathrm{T}}, \tag{2}$$

$$\mathbf{b} = (b_1, b_2, \ldots, b_k)^{\mathrm{T}}, \tag{3}$$

$$\mathbf{c} = (c_1, c_2, \ldots, c_l)^{\mathrm{T}}, \tag{4}$$

$$\mathbf{d} = (d_1, d_2, \ldots, d_n)^{\mathrm{T}}. \tag{5}$$

Furthermore, please note, that the total duration of each video fragment is around 30 min (the authors didn't intentionally trim data to have equal length). The following sliding window T_{WND} values of 0.5, 1, 5, 10, 15 and 20 seconds have been used for traffic features selection. This sliding window length is one of the dimensions for the features matrix, where each row represents traffic intensity samples of a single data window. The number of features (N_{C}, columns) and number of traffic segments (N_{R}, rows) for such a matrix can be defined as follows:

$$\frac{T_{\mathrm{WND}}}{T_{\mathrm{SMP}}} = N_{\mathrm{C}}, \tag{6}$$

$$\left\lfloor \frac{m}{N_{\mathrm{C}}} \right\rfloor = N_{\mathrm{R}}, \tag{7}$$

where N_{C} is the number of columns in the features matrix and N_{R} is the number of rows in the feature matrix calculated by floor rounding operation to get the lower integer value. According to the above definition, the traffic intensity vector is therefore reshaped into the features matrix with $N_{\mathrm{R}} \times N_{\mathrm{C}}$ dimensions. The first element is x_1, the last element in the first row will be the N_{C}th sample; the first element in the second row will be the $N_{\mathrm{C}+1}$th sample, and the last element in the whole matrix will be x_m, as it is shown by formula (8). Identically, we also need to reshape the traffic intensity vectors **a**, **b**, **c**, and **d** as the feature matrix.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{pmatrix},$$

$$\overline{\mathbf{reshape}} \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_{N_{\mathrm{C}}} \\ x_{N_{\mathrm{C}}+1} & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & x_m \end{bmatrix}. \tag{8}$$

According to the method mentioned above, we will get four different traffic intensity matrices after each intensity matrix reshaped procedure is finished: **A**, **B**, **C** and **D**. These matrices will have the same number of columns N_C expressed by (6). The next step is generation of merged features matrix with data from all classes by shuffling the rows of these 4 matrices in random order and also assigning the number as a label in labels vector. Assigning labels will be an essential step needed for supervised learning methods, that lets the classifier learn the row traffic comes from which video. The formula (9) shows and example of randomly generated features matrix **X** from reshaped intensity matrices **A**, **B**, **C**, and **D**. The square brackets [ ] in formula (9) denote vertical concatenation of matrices. At the same time, **Y** represents all merged vectors of different class labels:

- "0" label assigned for 720p@30FPS video quality network traffic (intensity matrix **A**);
- "1" label assigned for 720p@60FPS video quality network traffic (intensity matrix **B**);
- "2" label assigned for 1080p@30FPS video quality network traffic (intensity matrix **C**);
- "3" label assigned for 1080p@60FPS video quality network traffic (intensity matrix **D**).

Table 1. Deep learning classifiers

Classifier	Input dim	Conv layers	Hidden layers	Loss function	Optimizer	Epochs	Batch size	Verbose	Learning rate
DeepA	N_C	–	3	Sparse categorical crossentropy	Rmsprop	100	5	1	0.0001
CNN1	N_C	1	4	Categorical crossentropy	Adam	100	5	1	0.001
CNN2	N_C	3	4	Categorical crossentropy	Adam	100	5	1	0.001

X and **Y** make up a complete data set. Utilizing Python's library scikit-learn function train_test_split() function can realize training and testing data set splitting.

$$\mathbf{X} = \text{shuffle}\left(\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\\ \mathbf{C}\\ \mathbf{D}\end{bmatrix}\right) = \begin{bmatrix} b_1 & b_2 & b_3 & \cdots & b_{N_C} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N_{C+1}} & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & b_k \\ c_1 & c_2 & c_3 & \cdots & c_{N_C} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & d_n \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \quad \mathbf{Y} = \begin{pmatrix} 1 \\ \cdots \\ 0 \\ \cdots \\ 1 \\ 2 \\ \cdots \\ \vdots \\ 3 \\ \cdots \end{pmatrix}. \tag{9}$$

Training data set will be used to train deep learning classifiers. Multiple potential deep learning classifiers have been studied in the previous research, including DeepA, CNN1, and CNN2. The main differences between them are shown in Table 1, including the number of hidden layers, loss functions, optimizers, etc. The deep learning classifiers have significant advantages over the traditional machine learning classifiers, and convolutional neural networks excel even more at such classification problems than the deep learning classifiers without convolution layers.

For the classification task, accuracy is the most crucial indicator representing the proportion of samples with correct prediction over all samples, and it reflects the classifier's value and performance. F1-score considers the accuracy and recall of the classification model and can be regarded as a harmonic average of model accuracy and recall. Usually, accuracy and F1-score calculation convert multiclass classification tasks into multiple binary classification tasks, and the calculation steps are complicated. There are also some directly defined multiclass classification indicators. Jaccard similarity coefficient is an index to measure the similarity of two sets, that is, to calculate the similarity between two sets, and the element's value is 0 or 1. When the prediction result is utterly consistent with the actual situation, the coefficient is 1; when the prediction result is entirely inconsistent with the actual situation, the coefficient is 0; when the prediction result is a proper subset or a true superset of the actual situation, the distance is between 0 and 1. Cohen's kappa coefficient is also used to analyze the degree of agreement on classification problems [27]. Its range is between [–1, 1], with 1 indicating complete agreement; 0 or lower indicates inconsistency. We can get the overall performance on the testing set by averaging the predictions over all samples.

$$\mathrm{N} = \mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}, \tag{10}$$

$$\text{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{N}}, \tag{11}$$

$$\text{F1-score} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{12}$$

$$\text{Jaccard Index} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{13}$$

$$\text{Expected Accuracy} = \left(\frac{\mathrm{TP} + \mathrm{FP}}{\mathrm{N}}\frac{\mathrm{TP} + \mathrm{FN}}{\mathrm{N}}\right) + \left(\frac{\mathrm{TN} + \mathrm{FP}}{\mathrm{N}}\frac{\mathrm{TN} + \mathrm{FN}}{\mathrm{N}}\right), \tag{14}$$

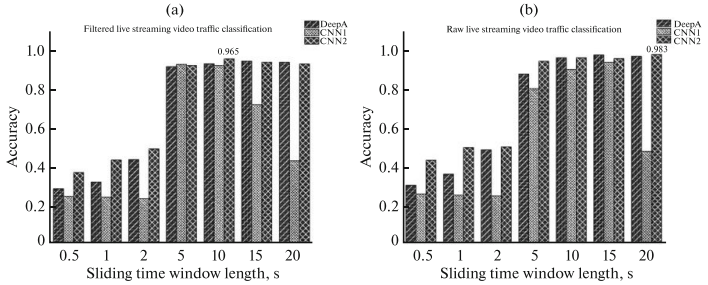


Fig. 2. (a) Multiple categories of filtered video traffic classification accuracy and (b) multiple categories of raw video traffic classification accuracy.

$$k = \frac{\text{Accuracy} - \text{Expected Accuracy}}{1 - \text{Expected Accuracy}}, \tag{15}$$

where TP, TN, FP, and FN mean true positive, true negative, false positive, and false negative, respectively. Together, they form all the samples of the dataset and are represented by N. Through the difference between predicted results and actual results label sets; we can define a variety of performance evaluation metrics, including formulas (11)–(13) and (15); k represents Cohen's kappa coefficient.

5. ANALYSIS OF CLASSIFICATION RESULTS

Figure 2a shows the multiple categories of filtered live streaming E-sports video traffic classification accuracy using previously defined deep learning classifiers. The time sampling interval is 10 ms, and the other variable is the sliding time window length. When the sliding time window length is below 5 s, all classifiers show poor classification accuracy of 0.5. If the sliding time window length is greater than or equal to 5 s, all classifiers can achieve over 0.9 classification accuracy. Still, the CNN1 classifier is not robust, as it degrades rapidly depending on the sliding time window length increased. CNN2 classifier obtains the highest classification accuracy for 10 s sliding time window length.

Figure 2b presents raw traffic classification accuracy when no IP filtering was performed for captured packets. If the sliding time window length is below 5 s, all classifiers only get around 0.5 classification accuracy. CNN2 classifier achieves over 0.9 classification accuracy when sliding time window length is greater than or equal to 5 s. Still, DeepA and CNN1 classifiers have an apparent difference with the CNN1 classifier in accuracy comparison terms; CNN2 also has a downtrend from increased sliding time window length. CNN2 classifier obtains around 100% accuracy for 20 s sliding time window length case. According to the actual performance of different classifiers, although the CNN model should have better performance than the typical deep learning classifier without convolution layers, the problem is caused by the kernel sizes used in the CNN1 classifiers. 500 × 500 sized large convolution kernel leads to significant information loss, while too small convolution kernel leads to too long computation time. Therefore, it is advisable to use multiple smaller convolution kernels rather than a large convolution kernel alone; the number of parameters and computational complexity are appreciably reduced when the connectivity is unchanged. Table 2 shows the detailed CNN2 configuration information, and it uses three convolution layers with small kernel sizes to minimize calculation parameters and obtain better generalization ability.

The difference between CNN1 and CNN2 classifiers is in the number of convolution layers and kernel size. After comparing DeepA and CNN2 classifiers, it has been determined that the performance difference is relatively small, using completely different parameters. Sparse categorical cross-entropy with digital coding and categorical crossentropy using one-hot coding, their loss functions are the same. DeepA uses the Rmsprop optimizer, and it relies on a global learning rate, which has been set to 0.0001, that is suitable for handling nonstationary targets. CNN2 uses Adam optimizer, which has a specific range after each iteration of the learning rate, making the parameters relatively stable. It has a small memory require-

Table 2. The description of CNN2 classifier

Layer	Type	Kernel size	Filter num	Stride	Pad	Output size
	Input	–	–	–	–	1 × 500
L1	1D Conv	5 × 5	100	1	2	500 × 100
	ReLU	–	–	–	–	–
L2	1D Conv	3 × 3	100	1	1	500 × 100
	ReLU	–	–	–	–	–
L3	1D Conv	1 × 1	100	1	0	500 × 100
	ReLU	–	–	–	–	–
L4	Flatten	–	–	–	–	50000
L5	Dense	50000 × 1	1000	–	–	1000 × 1
	ReLU	–	–	–	–	–
L6	Dense	1000 × 1	500	–	–	500 × 1
	ReLU	–	–	–	–	–
L7	Dense	500 × 1	250	–	–	250 × 1
	ReLU	–	–	–	–	–
L8	Dense	250 × 1	125	–	–	125 × 1
	ReLU	–	–	–	–	–
L9	Dense	125 × 1	4	–	–	4 × 1
	Softmax	–	–	–	–	–

ment and calculates different adaptive learning rates for various parameters. Similarly, more hidden layers can be used to fit nonlinear functions; it manifests CNN2 to have an apparent advantage over DeepA for a small sliding time window length. In general, CNN2 has the best performance in multiple categories of live streaming traffic classification tasks. Sliding time window length is also a crucial factor for classification performance. Still, after it reaches the value of 5 s, no further noticeable progress is observed for the classification accuracy.

In the test performed, the CNN2 classifier achieved the best performance for live streaming video traffic classification. However, some hyperparameters can still be utilized to improve classification performance. Usually, the size of the convolution kernel is specified, that is, its length and width. However, there is a third dimension of depth, which determines the number of filters. The number of channels of standard RGB scheme images is 3, and 1D-CNN is mainly used to extract features of time series data with only one dimension. The traffic data feature is a time series, so it does not have two dimensions; the depth dimension needs to be configured manually. Three convolutional layers in Table 2 used the same number of channels set to 100. The number of filters determines the number of channels output after convolution, so its impact on the classifier is significant.

Choosing a suitable batch size range is also an essential factor that affects the classifier's performance. If the batch size is too small, it will not have enough time to converge, and if the batch size is too large, the GPU will be out of memory; the generalization ability will also be insufficient. A larger batch size can calculate the gradient more accurately, and it is easier to achieve model convergence to the local optimum. Using a small batch size or even a single data training row is equivalent to artificially adding noise to make the model go out of the saddle point.

In the convolutional neural network, the random seed will be used as its initialization point, and initial weight needs to be chosen, or random gradient descent will be carried out when the Adam optimizer is applied. In the split function of the training set and testing set, random splitting will also lead to the different training set and testing set each time. These factors will affect each run of the same neural network model routine, and different results can be expected. Therefore, some regulations have been made during experiments to evaluate other hyperparameters, including batch size scale and number of filters influencing classification performance. The selected classifier is CNN2, with the best performance among all tested deep learning classifiers from Table 1. The sampling time interval is set to 10 ms, and the sliding time window length is 5 s. The tested number of filters includes values: 1, 10, 25, 50, 100, and 150; the

Table 3. Multiclass classification accuracy analysis with different batch sizes and number of filters

Running simes	Batch size:	Number of filters											
		1	10	25	50	100	150	1	10	25	50	100	150
		filtered traffic						raw traffic					
First sime	1	**0.23**	0.93	0.94	**0.22**	**0.22**	**0.22**	0.94	0.92	**0.22**	**0.23**	**0.22**	**0.22**
	5	**0.23**	0.94	0.93	0.95	0.92	0.95	**0.25**	0.93	0.92	0.95	0.94	0.93
	10	**0.23**	0.94	0.94	0.95	0.95	0.94	0.94	0.95	0.92	0.93	0.93	0.94
	16	0.96	0.95	0.95	0.95	0.95	0.95	0.93	0.93	0.94	0.94	0.94	0.94
	20	0.95	0.95	0.95	0.94	0.93	0.95	**0.25**	0.94	0.93	0.93	0.92	0.93
	32	0.93	0.95	0.95	0.94	0.94	0.95	0.88	0.92	0.92	0.92	0.93	0.93
Second time	1	0.92	0.92	0.94	0.92	**0.24**	**0.24**	**0.23**	0.95	0.94	0.93	**0.23**	**0.23**
	5	0.91	0.93	0.95	0.93	**0.24**	0.94	0.96	0.96	0.93	0.94	0.93	0.93
	10	**0.23**	0.93	0.95	0.95	0.95	0.95	0.94	0.95	0.93	0.93	0.93	0.95
	16	0.94	0.95	0.96	0.95	0.95	0.95	0.93	0.95	0.95	0.93	0.93	0.94
	20	**0.23**	0.93	0.94	0.96	0.96	0.95	**0.23**	0.96	0.95	0.93	0.95	0.94
	32	0.92	0.94	0.95	0.97	0.96	0.96	**0.23**	0.95	0.93	0.94	0.92	0.94
Third time	1	0.95	0.96	0.94	**0.21**	0.92	**0.21**	0.94	**0.24**	0.23	**0.23**	**0.23**	**0.23**
	5	**0.24**	0.96	0.95	0.94	0.93	0.93	**0.24**	0.95	0.92	0.95	0.93	0.92
	10	**0.24**	0.96	0.93	0.94	0.95	0.95	0.94	0.95	0.95	0.94	0.94	0.94
	16	0.96	0.97	0.94	0.94	0.94	0.94	0.93	0.95	0.94	0.95	0.94	0.94
	20	0.93	0.97	0.95	0.95	0.93	0.95	**0.24**	0.95	0.95	0.94	0.94	0.95
	32	0.95	0.96	0.95	0.95	0.94	0.94	**0.24**	0.95	0.94	0.94	0.94	0.95

batch size values were disposed to 1, 5, 10, 16, 20, and 32. These variable hyperparameters were matched in pairs to evaluate their impact on classification performance.

Table 3 shows the IP filtered live streaming traffic and raw traffic classification accuracy, respectively, using four different video quality traffic data sets previously. Due to the random nature of the deep learning classification results, a total of 3 training classifications have been performed, and the results were different each time. However, there were some similar rules. It has been determined that the batch size scale and number of filters have no linear relationship; the accuracy does not necessarily grow with the increase of one of these parameters. The highest classification accuracy achieved was 0.97, and some of the results were only around 0.25, which means they did not make any prediction, as all the results have the same label. Those worst results concentrate on only one filter, or the batch size is one. When batch size is equal to one, the number of samples selected for each training is one, stochastic gradient descent calculates one sample simultaneously, the gradient is inaccurate, the learning rate is reduced, and the model can hardly converge.

In the convolution process, channels are in the input layer should match with channels numbers in the convolution filter. In Table 2, the number of filters in the three convolutional layers is the same, and the output dimensions are also the same. Still, actual during the configuration process, the number of filters can be arbitrary. The number of filters determines the number of channels of the feature map after convolution, which is the contour details extracted in the convolution process. The number of filters, also named depth is one, then the feature map output by the layer is too tiny, which directly affects the classification result.

Table 4 shows the F1-score metric evaluated for IP filtered and raw traffic classification performance for different batch sizes and the numbers of filters. It shows a similar pattern as was for accuracy score in Table 3. There are some shallow F1-score values, for which the score was approximately 0.1, that is concentrated on only one filter or batch size is one; the highest F1-score achieved was over 0.96. The accuracy of some samples in Table 3 is very low, and the F1-score of these samples is also very low in Table 4, but the high classification accuracy is still accompanied by the high F1-score and average score reach to 0.95. This shows that F1-score closely correlates with accuracy. With the comprehensive evaluation index of

Table 4. Multiclass classification F1-score analysis with different batch sizes and number of filters

Running times	Batch size:	Number of filters											
		1	10	25	50	100	150	1	10	25	50	100	150
		filtered traffic						raw traffic					
First time	1	**0.09**	0.93	0.94	**0.09**	**0.09**	**0.09**	0.94	0.92	**0.09**	**0.10**	**0.09**	**0.09**
	5	**0.09**	0.93	0.93	0.94	0.91	0.94	**0.10**	0.93	0.92	0.95	0.95	0.93
	10	**0.09**	0.93	0.94	0.94	0.94	0.93	0.94	0.95	0.93	0.93	0.93	0.94
	16	0.96	0.95	0.95	0.94	0.94	0.94	0.93	0.93	0.94	0.94	0.94	0.94
	20	0.94	0.95	0.94	0.93	0.93	0.95	**0.10**	0.94	0.93	0.93	0.92	0.93
	32	0.93	0.94	0.95	0.94	0.94	0.94	0.88	0.92	0.93	0.92	0.93	0.93
Second time	1	0.93	0.92	0.94	0.92	**0.10**	**0.10**	**0.09**	0.94	0.93	0.92	**0.09**	**0.09**
	5	0.91	0.93	0.95	0.92	**0.10**	0.94	0.96	0.96	0.93	0.94	0.92	0.93
	10	**0.09**	0.93	0.95	0.94	0.95	0.95	0.94	0.95	0.93	0.93	0.93	0.95
	16	0.94	0.95	0.96	0.95	0.95	0.95	0.93	0.95	0.94	0.93	0.93	0.93
	20	**0.09**	0.93	0.94	0.96	0.96	0.95	**0.09**	0.95	0.95	0.93	0.94	0.93
	32	0.92	0.94	0.95	0.96	0.96	0.96	**0.09**	0.95	0.93	0.94	0.92	0.94
Third time	1	0.95	0.96	0.93	**0.09**	0.92	**0.09**	0.94	**0.10**	0.09	**0.10**	**0.09**	**0.09**
	5	**0.10**	0.96	0.95	0.94	0.93	0.93	**0.10**	0.95	0.91	0.94	0.92	0.92
	10	**0.10**	0.96	0.92	0.94	0.95	0.94	0.94	0.95	0.95	0.94	0.94	0.94
	16	0.96	0.96	0.94	0.93	0.93	0.94	0.93	0.95	0.94	0.95	0.94	0.94
	20	0.93	0.97	0.94	0.94	0.93	0.95	**0.10**	0.95	0.94	0.94	0.94	0.95
	32	0.94	0.96	0.95	0.95	0.94	0.94	**0.10**	0.95	0.94	0.94	0.94	0.94

Table 5. Multiclass Cohen's kappa coefficient analysis with different batch sizes and number of filters

Running times	Batch size:	Number of filters											
		1	10	25	50	100	150	1	10	25	50	100	150
		filtered traffic						raw traffic					
First time	1	**0.00**	0.91	0.92	**0.00**	**0.00**	**0.00**	0.92	0.90	**0.00**	**0.00**	**0.00**	**0.00**
	5	**0.00**	0.92	0.91	0.93	0.89	0.93	**0.00**	0.90	0.89	0.93	0.92	0.90
	10	**0.00**	0.91	0.92	0.93	0.93	0.92	0.92	0.93	0.90	0.91	0.91	0.92
	16	0.94	0.93	0.93	0.93	0.93	0.93	0.91	0.91	0.92	0.92	0.91	0.92
	20	0.93	0.93	0.93	0.92	0.91	0.94	**0.00**	0.92	0.90	0.91	0.89	0.91
	32	0.91	0.93	0.93	0.92	0.92	0.93	0.84	0.89	0.90	0.89	0.90	0.90
Second time	1	0.90	0.90	0.92	0.89	**0.00**	**0.00**	**0.00**	0.93	0.91	0.90	**0.00**	**0.00**
	5	0.88	0.90	0.94	0.90	**0.00**	0.92	0.95	0.95	0.90	0.92	0.90	0.90
	10	**0.00**	0.91	0.93	0.93	0.93	0.93	0.92	0.93	0.91	0.91	0.91	0.93
	16	0.92	0.93	0.95	0.94	0.94	0.94	0.91	0.94	0.93	0.91	0.91	0.92
	20	**0.00**	0.91	0.92	0.94	0.94	0.94	**0.00**	0.94	0.94	0.91	0.93	0.92
	32	0.90	0.92	0.94	0.95	0.95	0.95	**0.00**	0.94	0.91	0.92	0.90	0.92
Third time	1	0.93	0.94	0.92	**0.00**	0.90	**0.00**	0.91	**0.00**	0.00	**0.00**	**0.00**	**0.00**
	5	**0.00**	0.95	0.94	0.92	0.91	0.91	**0.00**	0.93	0.89	0.93	0.90	0.90
	10	**0.00**	0.95	0.90	0.92	0.94	0.93	0.92	0.93	0.94	0.92	0.92	0.92
	16	0.94	0.95	0.92	0.91	0.91	0.92	0.90	0.94	0.92	0.93	0.92	0.92
	20	0.90	0.96	0.93	0.93	0.91	0.94	**0.00**	0.93	0.93	0.92	0.92	0.93
	32	0.93	0.95	0.93	0.94	0.92	0.92	**0.00**	0.93	0.92	0.92	0.92	0.93

Table 6. Multiclass Jaccard similarity coefficient analysis with different batch sizes and number of filters

Running times	Batch size:	Number of filters											
		1	10	25	50	100	150	1	10	25	50	100	150
		filtered traffic						raw traffic					
First time	1	**0.06**	0.87	0.89	**0.05**	**0.05**	**0.05**	0.89	0.86	**0.06**	**0.06**	**0.06**	**0.06**
	5	**0.06**	0.88	0.87	0.89	0.85	0.89	**0.06**	0.87	0.85	0.90	0.90	0.87
	10	**0.06**	0.88	0.89	0.89	0.90	0.88	0.89	0.90	0.87	0.87	0.88	0.89
	16	0.92	0.90	0.90	0.89	0.89	0.89	0.88	0.88	0.89	0.89	0.88	0.89
	20	0.89	0.90	0.89	0.88	0.87	0.91	**0.06**	0.89	0.87	0.87	0.86	0.88
	32	0.87	0.90	0.90	0.89	0.88	0.89	0.79	0.85	0.86	0.86	0.87	0.87
Second time	1	0.87	0.86	0.89	0.85	**0.06**	**0.06**	**0.06**	0.90	0.88	0.86	**0.06**	**0.06**
	5	0.85	0.87	0.91	0.86	**0.06**	0.89	0.92	0.92	0.87	0.89	0.86	0.87
	10	**0.06**	0.88	0.91	0.90	0.90	0.90	0.88	0.90	0.87	0.87	0.87	0.90
	16	0.90	0.90	0.92	0.91	0.91	0.91	0.87	0.91	0.90	0.87	0.87	0.88
	20	**0.06**	0.88	0.89	0.92	0.92	0.91	**0.06**	0.92	0.91	0.87	0.90	0.88
	32	0.86	0.89	0.91	0.93	0.92	0.92	**0.06**	0.91	0.87	0.89	0.85	0.89
Third time	1	0.90	0.92	0.88	**0.05**	0.86	**0.05**	0.88	**0.06**	0.06	**0.06**	**0.06**	**0.06**
	5	**0.06**	0.92	0.91	0.89	0.87	0.87	**0.06**	0.90	0.84	0.90	0.86	0.86
	10	**0.06**	0.92	0.86	0.89	0.91	0.90	0.89	0.90	0.91	0.89	0.89	0.89
	16	0.92	0.93	0.89	0.87	0.88	0.89	0.87	0.91	0.89	0.90	0.89	0.89
	20	0.87	0.94	0.89	0.90	0.87	0.91	**0.06**	0.90	0.90	0.88	0.89	0.91
	32	0.90	0.92	0.90	0.91	0.89	0.89	**0.06**	0.90	0.89	0.89	0.89	0.90

recall and recall rate, there will be no uneven distribution of training samples or overfitting, which guarantees accuracy credibility.

Table 5 shows Cohen's kappa coefficient with IP filtered and raw traffic classification performance for different batch sizes and the numbers of filters. It shows a similar circumstance as the accuracy distribution in Table 3. For those classification accuracy achieved around 0.25 cases, Cohen's kappa coefficient only can get 0, which means this indicator is closely related to accuracy and is more stringent. The high classification accuracy also can obtain a high Cohen's kappa coefficient and more of them around 0.95, and it also verified the reliability of classification accuracy.

Table 6 shows the Jaccard similarity coefficient metric for IP filtered and raw live streaming traffic classification with different batch sizes and the numbers of filters. It is used to compare the similarity and differences between the limited sample sets. The larger the Jaccard similarity coefficient score, the higher the sample similarity between the predicted and actual values. Compared to Tables 3 and 4, it shows that when the classification accuracy and F1-score metric become poor, the Jaccard similarity coefficient score is also poor for the same samples. Like other evaluation metrics, those low Jaccard similarity coefficient score samples focus on one filter or batch size.

The lowest Jaccard similarity coefficient score is around 0.05, and the highest Jaccard similarity coefficient score can reach 0.94. However, compared with the F1-score, overall results have a 10% decrease using the Jaccard similarity coefficient score. The widespread score distribution conforms to the distribution of accuracy and F1-score, proving the performance guarantee for the training and testing sets.

6. CONCLUSIONS

This paper utilizes the direct traffic flows statistic data method to distinguish different quality E-sports live streaming videos. For videos of the same type and content, the difference in video quality resolution and frame rate per second has been successfully distinguished for four different types of videos: 720p@30FPS, 720p@60FPS, 1080p@30FPS, and 1080p@60FPS. The overall classification accuracy reaches 97%, the highest classification F1-score, Cohen's kappa coefficient, and the Jaccard similarity coefficient score close to 0.97, 0.96, and 0.94, respectively. Compared to three different deep learning clas-

sifiers, the CNN classifier has a significant advantage in multiclass classification tasks and shows its performance and robustness. Small convolution kernel size combined with multiple convolution layers also can bring more benefits. At the same time, the results of neural network operations are random, but some hyperparameters, such as training batch size and the number of filters, also significantly influence classification performance. Some cases should be avoided, such as when the batch size is one or only one filter has been used. This paper does not use the max pooling layer, because there is no feature explosion problem. Due to the relatively small number of samples, the CNN classifier also does not use the dropout layer, because there is no overfitting problem in experiments described in this paper. For large-scale neural networks, dropout can enhance the network's generalization ability, but it needs further evaluation.

FUNDING

This work has been supported by the European Regional Development Fund within the Activity 1.1.1.2 "Post-doctoral Research Aid" of the Specific Aid Objective 1.1.1 "To increase the research and innovative capacity of scientific institutions of Latvia and the ability to attract external financing, investing in human resources and infrastructure" of the Operational Program "Growth and Employment" (no. 1.1.1.2/VIAA/2/18/332).

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Wu, Z., Yao, T., Fu, Ya., and Jiang, Y.-G., Deep learning for video classification and captioning, in frontiers of multimedia research, *Front. Multimedia Res.*, Chang, S.-F., Ed., Association for Computing Machinery and Morgan & Claypool, 2017, pp. 3–29. https://doi.org/10.1145/3122865.3122867
2. Li, Yu., Po, L.-M., Cheung, Ch.-Ho, Xu, X., Feng, L., Yuan, F., and Cheung, K.-W., No-reference video quality assessment with 3D shearlet transform and convolutional neural networks, *IEEE Trans. Circuits Syst. Video Technol.*, 2016, vol. 26, no. 6, pp. 1044–1057. https://doi.org/10.1109/TCSVT.2015.2430711
3. Jang, H., Yang, H.-J., Jeong, D.-S., and Lee, H., Object classification using CNN for video traffic detection system, *21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, Mokpo, Korea (South), 2015, pp. 1–4. https://doi.org/10.1109/FCV.2015.7103755
4. Cricri, F., Roininen, M.J., Leppänen, J., Mate, S., Curcio, I.D.D., Uhlmann, S., and Gabbouj, M., Sport type classification of mobile videos, *IEEE Trans. Multimedia*, 2014, vol. 16, no. 4, pp. 917–932. https://doi.org/10.1109/TMM.2014.2307552
5. Sugano, T., Yamada, T., Sakazawa, Sh., and Hangai, S., Genre classification method for home videos, *IEEE Int. Workshop on Multimedia Signal Processing*, Rio de Janeiro, 2009, pp. 1–5. https://doi.org/10.1109/MMSP.2009.5293292
6. Wu, S., Ma, Yu-F., and Zhang, H.-J., Video quality classification based home video segmentation, *IEEE Int. Conf. on Multimedia and Expo*, Amsterdam, 2005, IEEE, 2005, p. 4. https://doi.org/10.1109/ICME.2005.1521399
7. Shanableh, T. and Ishtiaq, F., Pattern classification for assessing the quality of MPEG surveillance video, *Int. Conf. on Computer Systems and Industrial Informatics*, Sharjah, 2012, IEEE, 2012, pp. 1–5. https://doi.org/10.1109/ICCSII.2012.6454566
8. Horch, C., Habigt, J., Keimel, C., and Diepold, K., Evaluation of video quality fluctuations using pattern categorisation, *Sixth Int. Workshop on Quality of Multimedia Experience (QoMEX)*, Singapore, 2014, IEEE, 2014, pp. 117–122. https://doi.org/10.1109/QoMEX.2014.6982306
9. Tang, Sh., Li, Ch., Qin, X., and Wei, G., Traffic classification for mobile video streaming using dynamic warping network, *28th Wireless and Optical Communications Conf. (WOCC)*, Beijing, 2019, IEEE, 2019, pp. 1–5. https://doi.org/10.1109/WOCC.2019.8770669
10. Grabs, E., Petersons, E., Ipatovs, A., and Chulkovs, D., Supervised machine learning based classification of video traffic types, *24th Int. Conf. Electronics*, Palanga, Lithuania, 2020, IEEE, 2020, pp. 1–4. https://doi.org/10.1109/IEEECONF49502.2020.9141625
11. Yang, L.-Yu., Dong, Yu-N., Tian, W., and Wang, Z.-J., The study of new features for video traffic classification, *Multimedia Tools Appl.*, 2019, vol. 78, pp. 15839–15859. https://doi.org/10.1007/s11042-018-6965-6

12. Yang, L., Dong, Yu., Wu, Zh., Tang, P., and Feng, Yo., Feature mining for internet video traffic classification, *Int. Conf. on Network Infrastructure and Digital Content (IC-NIDC)*, Guiyang, China, 2018, IEEE, 2018, pp. 441–444. https://doi.org/10.1109/ICNIDC.2018.8525805
13. Islam, F.U., Liu, G., Zhai, J., and Liu, W., VoIP traffic detection in tunneled and anonymous networks using deep learning, *IEEE Access*, 2021, vol. 9, pp. 59783–59799. https://doi.org/10.1109/ACCESS.2021.3073967
14. Chen, Yi., Wang, W., Fu, L., and Zhang, X., Traffic model for HTTP video page, *Third Int. Conf. on Communications and Networking in China*, Hangzhou, China, 2008, IEEE, 2008, pp. 432–436. doi https://doi.org/10.1109/CHINACOM.2008.4685058
15. Dong, Yu., Yue, Q., and Feng, M., An efficient feature selection method for network video traffic classification, *IEEE 17th Int. Conf. on Communication Technology (ICCT)*, Chengdu, China, 2017, IEEE, 2017, pp. 1608–1612. https://doi.org/10.1109/ICCT.2017.8359902
16. Yuan, M. and Dong, Yu., Network video traffic classification using feature fusion, *IEEE 6th Int. Conf. on Computer and Communications (ICCC)*, Chengdu, China, 2020, IEEE, 2020, pp. 1847–1851. https://doi.org/10.1109/ICCC51575.2020.9345024
17. Dubin, R., Hadar, O., Richman, I., and Trabelsi, O., Dvir, A., and Pele, O., Video quality representation classification of Safari encrypted DASH streams, *Digital Media Industry & Academic Forum (DMIAF)*, Santorini, Greece, 2016, IEEE, 2016, pp. 213–216. https://doi.org/10.1109/DMIAF.2016.7574935
18. Takeshita, K., Kurosawa, T., Tsujino, M., Iwashita, M., Ichino, M., and Komatsu, N., Evaluation of HTTP video classification method using flow group information, *14th Int. Telecommunications Network Strategy and Planning Symp. (NETWORKS)*, Warsaw, 2010, IEEE, 2010, pp. 1–6. https://doi.org/10.1109/NETWKS.2010.5624929
19. Moreira, R., Rodrigues, L.F., Frosi Rosa, P., Aguiar, R.L., de Oliveira Silva, F., Packet Vision: A convolutional neural network approach for network traffic classification, *33rd SIBGRAPI Conf. on Graphics, Patterns and Images (SIBGRAPI)*, Porto de Galinhas, Brazil, 2020, IEEE, 2020, pp. 256–263. https://doi.org/10.1109/SIBGRAPI51738.2020.00042
20. A. V. Jain, Network traffic identification with convolutional neural networks, *IEEE 16th Int. Conf. on Dependable, Autonomic and Secure Computing, 16th Int. Conf. on Pervasive Intelligence and Computing, 4th Int. Conf. on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, Athens, Greece, 2018, IEEE, 2018, pp. 1001–1007. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00142
21. Zai-jian, W., Dong, Y., Shi, H., Lingyun, Y., and Pingping, T., Internet video traffic classification using QoS features, *Int. Conf. on Computing, Networking and Communications (ICNC)*, Kauai, Hawaii, 2016, IEEE, 2016, pp. 1–5. https://doi.org/10.1109/ICCNC.2016.7440599
22. Wu, Zh., Dong, Yu., Yang, L., and Tang, P., A new structure for internet video traffic classification using machine learning, *Sixth Int. Conf. on Advanced Cloud and Big Data (CBD)*, Lanzhou, China, 2018, IEEE, 2018, pp. 322–327. https://doi.org/10.1109/CBD.2018.00064
23. Zhang, H., Dong, L., Gao, G., Hu, H., Wen, Yo., and Guan, K., DeepQoE: A multimodal learning framework for video quality of experience (QoE) prediction, *IEEE Trans. Multimedia*, 2020, vol. 22, no. 12, pp. 3210–3223. https://doi.org/10.1109/TMM.2020.2973828
24. Zhai, F., Luna, C.E., Eisenberg, Y., Pappas, T.N., Berry, R., and Katsaggelos, A.K., Joint source coding and packet classification for real-time video transmission over differentiated services networks, *IEEE Trans. Multimedia*, 2005, vol. 7, no. 4, pp. 716–726. https://doi.org/10.1109/TMM.2005.850989
25. Li, H., Chen, Li, and Tian, J., A classification method for unbalanced surveillance video dataset, *13th IEEE Conf. on Industrial Electronics and Applications (ICIEA)*, Wuhan, China, 2018, IEEE, 2018, pp. 2821–2824. https://doi.org/10.1109/ICIEA.2018.8398190
26. Ghalut, T., Larijani, H., and Shahrabi, A., Content-based video quality prediction using random neural networks for video streaming over LTE networks, *IEEE Int. Conf. on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Liverpool, 2015, IEEE, 2015, pp. 1626–1631. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.245
27. Chicco, D., Warrens, M.J., and Jurman, G., The Matthews correlation coefficient (MCC) is more informative than Cohen's kappa and Brier score in binary classification assessment, *IEEE Access*, 2021, vol. 9, pp. 78368–78381. https://doi.org/10.1109/ACCESS.2021.3084050

[Publication 8]: Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks

Grabs, E., **Chen, T.**, Pētersons, E., Efrosinin, D., Ipatovs, A., Klūga, J., Čulkovs, D. Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks. In: 2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW 2021), Latvia, Riga, 7-8 October, 2021.

Features Extraction for Live Streaming Video Classification with Deep and Convolutional Neural Networks

Elans Grabs
Telecommunications Institute
Riga Technical University
Riga, Latvia

Tianhua Chen
Telecommunications Institute
Riga Technical University
Riga, Latvia

Ernests Petersons
Telecommunications Institute
Riga Technical University
Riga, Latvia

Dmitry Efrosinin
Institute of Stochastics
Johannes Kepler University
Linz, Austria

Aleksandrs Ipatovs
Telecommunications Institute
Riga Technical University
Riga, Latvia

Janis Kluga
Telecommunications Institute
Riga Technical University
Riga, Latvia

Dmitrijs Culkovs
Telecommunications Institute
Riga Technical University
Riga, Latvia

2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW) | 978-1-6654-2469-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/MTTW53539.2021.9607153

***Abstract* — These days, the high-quality live video streaming platforms are very widely used by lot of people all over the world. The demand for such services has increased even more in current pandemic conditions, so it is important to maintain certain level of Quality of Service for live streaming video. This article aims at pre-processing live streaming video-traffic packets for Deep Learning application of classification of different quality mode (Resolution and framerate) based only on statistical properties of network traffic Internet Protocol (IP) packets flow. The traffic intensity is used as a main traffic type descriptor with some additional transforms. The article presents some of classification accuracy analysis depending on multiple factors, such as: type of traffic pre-processing, length of traffic window, traffic intensity sampling time. The results show that there is a set of such parameters that leads to the best classification accuracy. This way, it may be possible for content operators (platforms) to provide differentiated services for their users. According to testing results, for Deep Learning application the direct traffic intensity samples allows achieving good accuracy of 94% and higher without any additional transforms. This, of course, depends on the classifier model parameters, and article provides insight into them as well.**

Keywords — Features Extraction, Deep Learning, Convolutional Neural Networks, Network Traffic, Multi-class Classification

I. Introduction

Nowadays, the Machine Learning approach, and more specifically, Deep Learning Neural Networks (DNN), can be used in almost any aspect of our daily lives. This includes network traffic applications as well [1], [2]. The authors of the presented article are working in a field of video traffic classification by using Deep Neural Networks. This idea of video traffic classification itself is not novel and has been already studied from many points of view [3], [4]. It is very important to perform correct selection of traffic features [5], [6], [7]. Mostly, these methods use some protocol information and/or statistical parameters of traffic flows. For more details on author's previous research, as well as data collection and preparation, please see [8]. The mentioned article was focused on binary classification problem between live streaming (real-time) video and on-demand video (record) playback classification, very similar to task studied in [9] for mobile networks video traffic. In the presented research, however, the main objective is to perform multi-class classification of video-traffic quality mode, with a total number of 4 possible quality settings. As it was previously mentioned, there is a lot of research going on for this topic, and there are different approaches for performing such a classification. For example, [10] uses actual video content, extracted by Convolutional Neural Networks (CNN) to train Deep Learning model to perform this classification. At the same time, [11] proposes to introduce unsupervised machine learning at the server side and Deep Learning at the client side. For the current research of the presented article, however, the traffic intensity was chosen as a main feature for classification. Such approach has also been used, for example for World-Wide Web (WWW) traffic classification in [12]. Furthermore, this traffic intensity can be processed by using different transforms or statistical methods to improve efficiency.

This work has been supported by the European Regional Development Fund within the Activity 1.1.1.2 "Post-doctoral Research Aid" of the Specific Aid Objective 1.1.1 "To increase the research and innovative capacity of scientific institutions of Latvia and the ability to attract external financing, investing in human resources and infrastructure" of the Operational Programme "Growth and Employment" (No.1.1.1.2/VIAA/2/18/332)

II. Simulation Model Description

The primary purpose of the experiment described in the presented article was to find the most suitable traffic pre-processing procedure for generating features. Afterwards, these traffic features can be used for DNN/CNN model training and accuracy evaluation.

As it was mentioned in previous section, there are multiple possible options for features selection, however in this research the focus has been made on using properties of network traffic packets flow as stochastic process. This means, that no additional data from any protocol level is used, except for the number of received packets from specific

connection (or, as it will be shown later, it is possible to use raw, i.e., unfiltered traffic, as well).

The simplified flowchart of entire data collecting, pre-processing and training/testing simulation is displayed in Fig. 1. below.

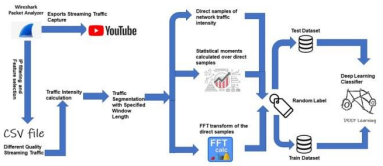


Fig. 1. The flowchart of experiment and data preparation procedure.

The source of network traffic packets was Youtube platform, where live streams related to League of Legends contests have been going on. These live streams were played back on laptop computer connected via IEEE 802.11n standard to Internet, and at the same time the traffic packets were captured by network monitoring tool WireShark.

The live video streams were captured with different video quality mode during the playback of a duration 30 minutes per video fragment:

1. 720p@30FPS,
2. 720p@60FPS,
3. 1080p@30FPS,
4. 1080p@60FPS.

The dumps of captured data include all information about network traffic and live video streams; however, the volume of data is just too big, especially because packet payload was of no interest. Therefore, the first step is to reduce complexity of data by selecting traffic features. At this step different mathematical transforms can also be used. In research being described in the presented article, only the packet occurrence was fixed as an event in exported file, along with packet reception time and packet size (currently not being used for any purpose).

The next step is optional IP filtering. In experiment described in the presented article, all packets received by laptop have been captured, including operating system background processes, messengers, etc. It is true, that their total load is negligible in comparison with High-Definition (HD) or Full HD streaming video playback, however for clean results only live streaming video related packets can be saved, by performing source/destination IP address filtering. After this step, the data is exported to Comma-Separated Values (CSV) file format.

The mentioned above data is then exported to 4 CSV files, one per each traffic quality mode, which can be used with any programming language. For this project, as for many others, Python programming language has been used with TensorFlow and Keras framework for Deep Learning models. All the required software was set up on a computer equipped with Nvidia RTX 3060 Graphical Processing Unit (GPU) for faster calculations and batch processing script has been written to train and test accuracy of different (specified by user) models in different scenarios.

The batch-processing script is not limited to 4 data classes, and the number of classes is defined only by number of input CSV files. These CSV files are being merged and shuffled, with automatic label assignment as an integer number: 0, 1, 2, … Then, the dataset is split into training dataset and testing dataset. Please note, that in this particular case, each class is represented by 30 minutes long video fragment, so the dataset is well-balanced.

The batch-processing script outputs results as specified metrics for each simulated scenario. Different performance metrics can be specified, such as Accuracy, Precision, Recall, F1-score and any other, included in Python libraries. Results presented in the article have been shown for Accuracy only, which is evaluated as:

$$Accuracy = \frac{N^+}{N_{total}}, \tag{1}$$

where N^+ is the number of total correct predictions of testing entries, and N_{total} is the total number of testing entries.

III. Data Preprocessing and Features Selection

After capturing data packets with WireShark and exporting results into CSV file format, the preparation of dataset can begin.

During the research of authors, it has been determined, that network traffic intensity is a very promising parameter, as it describes network traffic over specific time intervals in average terms. Furthermore, it has been long ago proven, that modern internet multimedia traffic is self-similar [13], [14], and complex enough DNN/CNN model may be able to extract the long-term dependency of such a traffic and use it for classification.

Hence, the first important parameter affecting classification accuracy is traffic intensity time interval T_{SMP}. However, the selection of this interval is not straightforward, and multiple considerations must be taken into account:

- The higher value of sampling time interval allows collecting statistics of packets over longer time interval and take into account long-range dependence of network traffic;
- At the same time, the higher value of sampling time interval means, the data needs to be collected for specific time interval and the update speed of traffic will be slower, which can degrade performance of classification, especially in case of bursty ON-OFF type traffic.

This problem can be partially alleviated by applying additional sliding window with a duration of T_{WND} over traffic intensity samples. This way, it is possible to have traffic samples with frequency supported by system and at the same time define arbitrary long window length for capturing long-range dependency properties.

Furthermore, the number of features of training/testing dataset entry depends on both time intervals and represents the number of columns for features matrix, which can be evaluated as follows:

$$N_C = \frac{T_{WND}}{T_{SMP}}, \quad (2)$$

where T_{WND} is the sliding window length time and T_{SMP} is the traffic intensity sampling time. Then, the number of such rows (entries) of features matrix will be, accordingly:

$$N_R = \frac{N}{N_C}, \quad (3)$$

where N is the total number of traffic intensity samples calculated for specific data-file.

Furthermore, the intensity samples stored in window length can be altered by using different transforms and/or statistical methods. For example, in [15] Discrete Cosine Transform was used to present features data in more compact and descriptive form, whereas [16] describes multiple approaches, including Discrete Wavelet Transform. Considering results of the previous research of the authors in [8], where traffic intensity samples over window were replaced by specific number of statistical moments, showed improved results. However, in that research only simple (binary) classifiers were studied, without Deep Learning models.

So, to determine the best features representation type, multiple options must be studied, such as:

- Direct samples of network traffic intensity over the window with time duration of T_{WND};
- Fast Fourier Transform (FFT) of the direct samples, over the same T_{WND} time duration. This duration needs to be correctly chosen to provide valid number of samples for FFT algorithm and it affects directly the number of FFT points according to (2);
- Statistical moments calculated over direct samples within the window with duration of T_{WND}. Multiple statistical moments can be calculated, for example in this research the maximum number of such moments is set to $K_{\mathrm{MAX}} = 6$.

Finally, the traffic intensity sampling time interval T_{SMP} and features selection window time interval T_{WND} also affects the performance of classifier. Multiple combinations of these parameters were studied, and accuracy has been evaluated for the following ranges of parameters:

- Intensity time values T_{SMP}:

  1 ms, 10 ms, 100 ms
- Traffic window time values T_{WND}:

  0.5 s, 1 s, 2 s, 5 s, 10 s

This way, there are total of 15 different pairs of the parameters specified in ranges above. In addition, there are 3 different features methods, so there will be total of 45 classification experiments. For each of these experiments, a model is being trained and accuracy is evaluated. Please note, however, that for each model the hyperparameters were set to be the same, and only the number of inputs can be different, depending on the number of traffic features.

IV. Selected Models and Parameters

Before conducting experiments, multiple possible models need to be defined. This research is focused on DNN and CNN, so multiple artificial neural networks must be defined: their architecture and hyperparameters.

During the search for fitting models, multiple candidates have been determined. In this article the following models will be described and tested:

- Deep Neural Network with 3 hidden layers (***DeepModel1A***), see Table I for details;
- Convolutional Neural Network with 1 convolutional layer and 4 hidden layers (***CNN***), see Table II for details;
- Convolutional Neural Network with 3 convolutional layer and 4 hidden layers (***CNN2***), see Table III for details.

TABLE I: DeepModel1A classifier architecture and parameters.

DeepModel1A **Model**	**Details**
Hidden Layer 1	500 nodes, ReLu activation
Hidden Layer 2	250 nodes, ReLu activation
Hidden Layer 3	125 nodes, ReLu activation
Output Layer	SoftMax activation
Model parameters:	Optimizer: RMSProp, rate 0.0001 Number of epochs: 100 Batch Size: 5 Loss function: Sparse Categorical Crossentropy

TABLE II: CNN classifier architecture and parameters.

CNN **Model**	**Details**
Conv1D layer	100 filters, kernel size N_c, Sigmoid activation
Flatten layer	No parameters
Hidden Layer 1	1000 nodes, Sigmoid activation
Hidden Layer 2	500 nodes, Sigmoid activation
Hidden Layer 3	250 nodes, Sigmoid activation
Hidden Layer 3	125 nodes, Sigmoid activation
Output Layer	SoftMax activation
Model parameters:	Optimizer: Adam Number of epochs: 100 Batch Size: 5 Loss function: Sparse Categorical Crossentropy

TABLE III: CNN2 classifier architecture and parameters.

CNN2 **model**	**Details**
Conv1D layer	100 filters, kernel size 5, ReLu activation
Conv1D layer	100 filters, kernel size 3, ReLu activation
Conv1D layer	100 filters, kernel size 1, ReLu activation
Flatten layer	No parameters
Hidden Layer 1	1000 nodes, ReLu activation
Hidden Layer 2	500 nodes, ReLu activation
Hidden Layer 3	250 nodes, ReLu activation
Hidden Layer 3	125 nodes, ReLu activation
Output Layer	SoftMax activation
Model parameters:	Optimizer: Adam Number of epochs: 100 Batch Size: 5 Loss function: Sparse Categorical Crossentropy

Each of the models described in Table I to Table III has been trained with all possible datasets obtained at the data preprocessing stage, by pairing up different traffic intensity time intervals T_{SMP} and sliding window length time T_{WND}. At this stage no additional search was made to improve accuracy via model hyperparameters, such as batch size or number of filters for CNN models.

V. Analysis of Results

After performing training of models described in previous section, the evaluation of accuracy has been performed in all modeled scenarios. The total number of performed experiments was considerably large to provide all results in this article, so only most important of obtained results will be presented in this section.

As it was previously mentioned, there were 3 models under training/testing process: DNN model ***DeepModel1A*** and two CNN models – ***CNN*** and ***CNN2***, accordingly. Each of these models was trained and tested with different data preprocessing modes by changing the following traffic parameters:

- Traffic intensity sampling time T_{SMP};
- Traffic window length time T_{WND}.

Hence, the results obtained after running batch-processing scripts were formatted as tables, and examples of such tables can be seen below in Table IV, Table V and Table VI for direct data processing, FFT and statistical modes, accordingly. All these tables show the accuracy metric for different window length assuming the sampling time is $T_{SMP} = 10$ ms.

TABLE IV: Direct Traffic processing mode results for 10 ms sampling time (IP Filtering enabled)

Sliding Window Length	Classification Accuracy		
	DeepModel1A	CNN	CNN2
0.5 seconds	0.319	0.248	0.426
1 second	0.372	0.244	0.466
2 seconds	0.443	0.241	0.532
5 seconds	0.909	0.820	0.931
10 seconds	0.900	0.909	0.948

TABLE V: FFT Traffic processing mode results for 10 ms sampling time (IP Filtering enabled)

Sliding Window Length	Classification Accuracy		
	DeepModel1A	CNN	CNN2
0.5 seconds	0.316	0.249	0.454
1 second	0.443	0.242	0.523
2 seconds	0.468	0.254	0.582
5 seconds	0.645	0.700	0.677
10 seconds	0.519	0.700	0.667

TABLE VI: Statistical Traffic processing mode results for 10 ms sampling time (IP Filtering enabled). Number of moments is 4.

Sliding Window Length	Classification Accuracy		
	DeepModel1A	CNN	CNN2
0.5 seconds	0.397	0.244	0.407
1 second	0.356	0.242	0.448
2 seconds	0.487	0.242	0.478
5 seconds	0.656	0.216	0.673
10 seconds	0.680	0.550	0.697

Please, note that for statistical data processing results in Table VI the number of statistical moments is 4. The simulation results were obtained for each window length over a range of number of statistical moments from 1 to 6, however for simpler representation only results for number of moments equal to 4 have been shown in Table VI. The change of number of statistical moments leads to only slight changes of accuracy, as can be verified from results in Table VII, where accuracy results are presented for different number of statistical moments for a fixed window length $T_{WND} = 10$ sec.

TABLE VII: Statistical Traffic processing mode results for 10 ms sampling time and window length time 10 sec (IP Filtered).

Number of Statistical Moments	Classification Accuracy		
	DeepModel1A	CNN	CNN2
1	0.597	0.554	0.580
2	0.675	0.571	0.671
3	0.688	0.558	0.684
4	0.671	0.550	0.700
5	0.662	0.554	0.658
6	0.680	0.550	0.749

The results in Table VII clearly show, that up to certain number of statistical moments there is accuracy improvement for all three tested models, and the number of such moments mostly is 3 or 4. On very rare occasions, there is further improvement up to 6 moments. The most significant difference, however, is observed when changing number of statistical moments from 1 to 2.

Overall, analysis of obtained results in Table IV, Table V and Table VI shows, that the highest accuracy is achieved when no special techniques were used for features extraction. This, mostly, is because DNN and CNN models have been used, which intrinsically perform deep analysis of time series looking for the most outstanding features. Therefore, the further visualized results in this section will be provided for direct traffic processing mode.

For easier interpretation of obtained results, 3D surface plots can be constructed. To get more accurate data, additional experiments were performed for the following ranges of traffic parameters:

- Window length time values: 0.5s, 1s, 2s, 5s, 10s, 15s, 20s;
- Intensity sampling time values: 10ms, 25ms, 50ms, 100ms, 200ms.

The results were obtained both for IP filtered traffic data, shown in Fig. 2-4., and for Raw traffic data (with packets from all IP addresses) shown in Fig. 5-7.

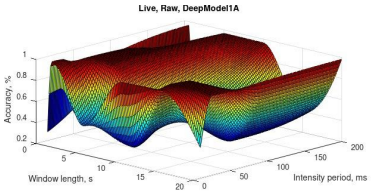


Fig. 2. Accuracy 3D surface plot for live streaming video traffic classification without IP filtering of packets for ***DeepModel1A.***

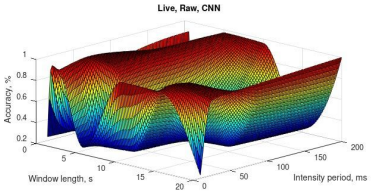


Fig. 3. Accuracy 3D surface plot for live streaming video traffic classification without IP filtering of packets for ***CNN*** model.

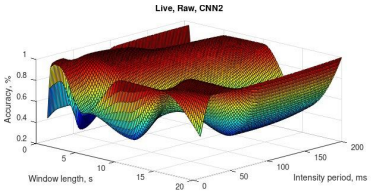


Fig. 4. Accuracy 3D surface plot for live streaming video traffic classification without IP filtering of packets for ***CNN2*** model.

As it can be seen from comparison of the results presented in form of 3D surface plots in Fig. 2. – Fig. 7., it is possible to achieve very high accuracy in all cases, furthermore the shape of all 3D surfaces is very similar, although for each experiment a new training process was conducted.

This means, that the solution indeed converges to some optimum set of neural network's weights, depending on traffic parameters – window length or intensity sampling time, regardless of whether traffic has been IP filtered or not. Please note, however, that during the traffic capturing the number of background processes was kept minimal. So, additional studies are necessary for case of significant background traffic during the process of live stream video traffic capture.

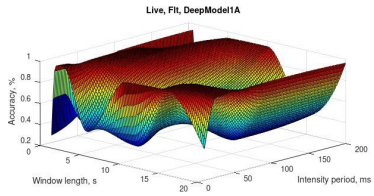


Fig. 5. Accuracy 3D surface plot for live streaming video traffic classification with IP filtering of packets for ***DeepModel1A.***

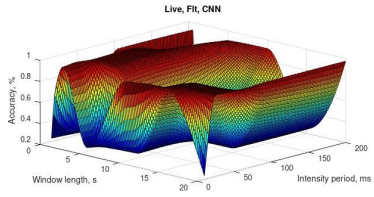


Fig.6. Accuracy 3D surface plot for live streaming video traffic classification with IP filtering of packets for ***CNN*** model.

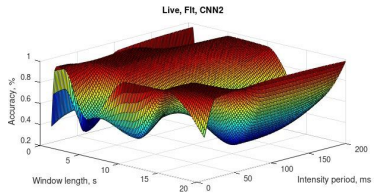


Fig. 7. Accuracy 3D surface plot for live streaming video traffic classification with IP filtering of packets for ***CNN2*** model.

CONCLUSION

This article presents some of the results obtained during the research on accuracy evaluation for live streaming video traffic classification with different selection of features.

Based on results obtained during the simulation, it has been determined that for DNN/CNN models the best selection of features are plain intensity samples without additional transforms/statistical analysis. The accuracy difference is significant, and it seems that CNN model searches for hidden pattern in data, which is lost after transforms. The discrete Wavelet transform (DB-3 base wavelet, 2 and 3 levels of filter banks) also didn't show improvement of accuracy compared to FFT method, although the results weren't presented in this article.

The comparison of different DNN/CNN models architecture shows, that CNN models perform better due to presence of 1D convolutional layers, which are used to detect long-range dependence of self-similar network traffic.

The comparison of ***CNN*** and ***CNN2*** models shows, that Rectified Linear Unit (ReLU) activation function is generally better suited for the problem studied in presented article.

Finally, the comparison of Fig. 2. and Fig. 5., Fig. 3. and Fig. 6., Fig. 4. and Fig. 7., shows, that the influence of IP filtering is minimal, if there are no background processes generating significant amount of traffic.

The future work is related to testing obtained classifiers with new additional data captured from another computer and, perhaps, network connection. Furthermore, the model of traffic generation needs to be determined to intentionally modify data and study the effects of such anomalies on performance of the classifiers. Finally, the research can continue on finding even better classifiers based on modern approaches, such as Long Short-Term Memory (LSTM) networks and/or Ensemble models.

REFERENCES

[1] Boutaba, R., Salahuddin, M.A., Limam, N. et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J Internet Serv Appl 9, 16 (2018). https://doi.org/10.1186/s13174-018-0087-2

[2] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," in IEEE Communications Surveys & Tutorials, vol. 10, no. 4, pp. 56-76, Fourth Quarter 2008, doi: 10.1109/SURV.2008.080406.

[3] Z. Wu, Y. Dong, L. Yang and P. Tang, "A New Structure for Internet Video Traffic Classification Using Machine Learning," 2018 Sixth International Conference on Advanced Cloud and Big Data (CBD), 2018, pp. 322-327, doi: 10.1109/CBD.2018.00064.

[4] R. Andersson, 'Classification of Video Traffic.: An Evaluation of Video Traffic Classification using Random Forests and Gradient Boosted Trees', Dissertation, 2017.

[5] Yang, LY., Dong, YN., Tian, W. et al. The study of new features for video traffic classification. Multimed Tools Appl 78, 15839–15859 (2019). https://doi.org/10.1007/s11042-018-6965-6

[6] W. Zai-jian, Y. Dong, H. Shi, Y. Lingyun and T. Pingping, "Internet video traffic classification using QoS features," 2016 International Conference on Computing, Networking and Communications (ICNC), 2016, pp. 1-5, doi: 10.1109/ICCNC. 2016.7440599.

[7] Y. Dong, Q. Yue and M. Feng, "An efficient feature selection method for network video traffic classification," 2017 IEEE 17th International Conference on Communication Technology (ICCT), 2017, pp. 1608-1612, doi: 10.1109/ICCT.2017.8359902.

[8] E. Grabs, E. Petersons, A. Ipatovs and D. Chulkovs, "Supervised Machine Learning based Classification of Video Traffic Types," 2020 24th International Conference Electronics, 2020, pp. 1-4, doi: 10.1109/IEEECONF49502.2020.9141625.

[9] S. Tang, C. Li, X. Qin and G. Wei, "Traffic Classification for Mobile Video Streaming Using Dynamic Warping Network," 2019 28th Wireless and Optical Communications Conference (WOCC), 2019, pp. 1-5, doi: 10.1109/WOCC.2019.8770669.

[10] Varga, D., Szirányi, T. No-reference video quality assessment via pretrained CNN and LSTM networks. SIViP 13, 1569–1576 (2019). https://doi.org/10.1007/s11760-019-01510-8

[11] M. T. Vega, D. C. Mocanu, J. Famaey, S. Stavrou and A. Liotta, "Deep Learning for Quality Assessment in Live Video Streaming," in IEEE Signal Processing Letters, vol. 24, no. 6, pp. 736-740, June 2017, doi: 10.1109/LSP.2017.2691160.

[12] W. W. Vithanage and A. S. Atukorale, "A novel classifier for engineering web traffic," 2011 IEEE Symposium on Computers and Communications (ISCC), 2011, pp. 1009-1016, doi: 10.1109/ISCC.2011.5983974.

[13] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of Ethernet traffic (extended version). IEEE/ACM Transactions on Networking, 2:1–15, 1994.

[14] O. Lazaro, D. Girma and J. Dunlop, "Statistical analysis and evaluation of modelling techniques for self-similar video source traffic," 11th IEEE International Symposium on Personal Indoor and Mobile Radio Communications. PIMRC 2000. Proceedings (Cat. No.00TH8525), 2000, pp. 1540-1544 vol.2, doi: 10.1109/PIMRC.2000.881676.

[15] Saad, M.A., Bovik, A.C., Charrier, C.: Blind prediction of natural video quality. IEEE Trans. Image Process. 23(3), 1352–1365 (2014)

[16] Abbasi, M., Shahraki, A., & Taherkordi, A. (2021). Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey. Comput. Commun., 170, 19-41.

Tianhua Chen was born in Nanjing, China, in 1995. He received his Master's degree in Engineering in Telecommunications (2021) from Riga Technical University. He has worked as a network engineer at multiple Internet Service Providers. Since 2023, he has been a researcher at Riga Technical University. His research interests are in multimedia, XR application network traffic analysis, as well as 5G Open RAN and SDN application development.