Leveraging AI with Wavelet-Transformed Trend Features for Low-Latency Internet Traffic Identification

Ramazan Enisoglu

School of Science and Technology
Department of Engineering
City St George's, University of London

Supervisor: Dr. Veselin Rakocevic

February 2025

This dissertation is submitted for the degree of

Doctor of Philosophy

This thesis is dedicated to Allah the Almighty, my most cherished ones: my wife and daughter, Gamze & Duru; my beloved parents, Mr. Bayram Ali & Mrs. Fatma; and my sister and brother, Emriye & Furkan.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Veselin Rakocevic, for his invaluable guidance, support, and continuous encouragement throughout this journey. His expertise and insight have been instrumental in shaping the direction of my research and in the completion of this dissertation.

I extend my sincere thanks to the School of Science and Technology at City, University of London, for providing the resources and environment necessary for my academic endeavors. I am particularly grateful to my colleagues and fellow researchers who have provided critical feedback, collaboration, and camaraderie during my studies. A special mention goes to my friend Burak Alp Inan, who, like me, has been a recipient of the prestigious YLSY program by the Ministry of Education of the Republic of Turkey. His friendship and shared experiences have been invaluable throughout this journey.

I would also like to acknowledge the unwavering support of my family, particularly my parents, Mr. Bayram Ali and Mrs. Fatma, and my sister and brother, Emriye and Furkan, whose love and encouragement have been a constant source of strength. A special thanks to my wife, Gamze, for her understanding and support, and to my daughter, Duru, whose presence has brought joy and motivation to my life during this time.

Lastly, I gratefully acknowledge the opportunity provided by the YLSY pro-

gram, which has enabled me to pursue my studies abroad. This program, through a rigorous selection process, has not only supported me financially but has also made it possible for me to achieve my academic goals on an international level.

Abstract

In the rapidly evolving landscape of digital communications, the demand for fast, reliable, and efficient network traffic management has become more critical than ever. Traditional Internet traffic classification methods, such as payload-based or port-based techniques, are inadequate in the face of encrypted traffic and dynamic port assignments. Addressing these challenges, this thesis presents a comprehensive approach to improving Internet traffic classification by leveraging Artificial Intelligence (AI) with wavelet-transformed trend features, thereby paving the way for more adaptive and accurate solutions in modern network environments.

The core of this research lies in the development of advanced feature extraction techniques using Wavelet Transforms combined with trend analysis. Wavelet Transforms capture both the time and frequency domain characteristics of network traffic, providing a comprehensive view of traffic behavior. By integrating these features with AI models, particularly Machine Learning and Deep Learning algorithms, the thesis enhances the classification accuracy of low-latency traffic, which is critical for applications such as video conferencing, online gaming, and real-time financial transactions. This work contributes to the ongoing efforts to enhance network performance and reliability, ultimately supporting the growing demands of digital applications in today's interconnected

world.

This work also introduces the integration of trend-based features, which reflect the periodicity, seasonality, and long-term behavior of network traffic. These features, when combined with wavelet-transformed data, enhance the robustness of the AI model, making it more resilient to the dynamic nature of Internet traffic. The findings of this research have substantial implications for the design of next-generation network management systems, enabling more efficient allocation of resources and better Quality of Service (QoS) for latency-sensitive applications.

Another considerable contribution of this research is the creation of a dataset and the implementation of an AI algorithms that can dynamically adapt to changing network traffic patterns. Trained on wavelet-transformed trend features, the proposed model captures the unique characteristics of low-latency traffic even in mixed and complex environments. Extensive experiments using real network traffic data to validate the effectiveness of the proposed methods, demonstrating significant improvements in classification accuracy compared to traditional methods.

Keywords: *Low-latency Internet traffic, Wavelet Transform (WT), Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL), Network traffic classification, Trend analysis, Feature extraction, Real-time traffic identification, Quality of Service (QoS)*

Contents

List of Figures

List of Tables

Acronyms

ACF Autocorrelation Function.

AE Auto Encoder.

AI Artificial Intelligence.

ANN Artificial Neural Networks.

API Application Programming Interface.

AR Autoregressive.

AUP Acceptable Use Policy.

BPF Berkeley Packet Filter.

C4.5 Extension of Iterative Dichotomiser 3 Algorithm.

CART Classification and Regression Tree.

CDN Content Delivery Networks.

CFS Correlation-based Feature Selection.

CNN Convolutional Neural Network.

CON Consistency-based Subset Search.

CSF Cybersecurity Framework.

CWT Continuous Wavelet Transform.

DA3 The average throughput in downstream over 3 sec.

DASH Dynamic Adaptive Streaming over HTTP.

DL Deep Learning.

DNN Deep Neural Network.

DNS Domain Name System.

DPDK Data Plane Development Kit.

DPI Deep Packet Inspection.

DT Decision Tree.

DU D to U ratio.

EB Exabyte.

EMA Exponential Moving Average.

FCS Frame Check Sequence.

FPGA Field-Programmable Gate Array.

FT Fourier Transfrom.

FTP File Transfer Protocol.

GDPR General Data Protection Regulation of the European Union.

GPU Graphics Processing Unit.

GR Gain Ratio.

HLS HTTP Live Streaming.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IG Information Gain.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

IoT Internet of Things.

IP Internet Protocol.

IPFIX Internet Protocol Flow Information Export.

IPv4 Internet Protocol version 4.

IPv6 Internet Protocol version 6.

k-NN k-Nearest Neighbors.

MA Moving Average.

ML Machine Learning.

MLP Multilayer Perceptron.

MRA Multi Resolution Analysis.

MTU Maximum Transmission Unit.

NIC Network Interface Card.

NLANR National Laboratory for Applied Network Research.

OSI Open Systems Interconnection.

P2P Peer-to-Peer.

PACF Partial Autocorrelation Function.

PDF Probability Distribution Function.

PPS Packets Per Second.

QoS Quality of Service.

RDA3 Throughput ratio (slope) in downstream.

RF Random Forest.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

RTMP Real-Time Messaging Protocol.

RUA3 Throughput ratio (slope) in upstream.

SGD Stochastic Gradient Descent.

SIP Session Initiation Protocol.

SMA Simple Moving Average.

SMTP Simple Mail Transfer Protocol.

SSL Secure Sockets Layer.

STFT Short Time Fourier Transform.

SVM Support Vector Machine.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TTL Time to Live.

UA3 The average throughput in upstream over 3 sec.

UDP User Datagram Protocol.

VoIP Voice Over IP.

WMA Weighted Moving Average.

WT Wavelet Transform.

CHAPTER 1

Introduction

This chapter introduces the context and motivation for the research, explaining the challenges in network traffic classification and the rationale for adopting advanced techniques such as Wavelet Transform and Artificial Intelligence. It also outlines the research objectives, contributions, and a comprehensive structure of the thesis, providing a roadmap for understanding how each component addresses these challenges.

1.1 Background and Motivation

The growth in Internet usage, fueled by advances in online services, streaming platforms, and cloud computing, has significantly increased the complexity of network traffic management. Figure 1.1 Ericsson (2021) illustrates the rapid increase in global mobile network data traffic. Ensuring high service quality amidst such growth is critical for user satisfaction, particularly for latency-sensitive applications such as online gaming, video conferencing, and real-time financial transactions, where delays can severely degrade performance.

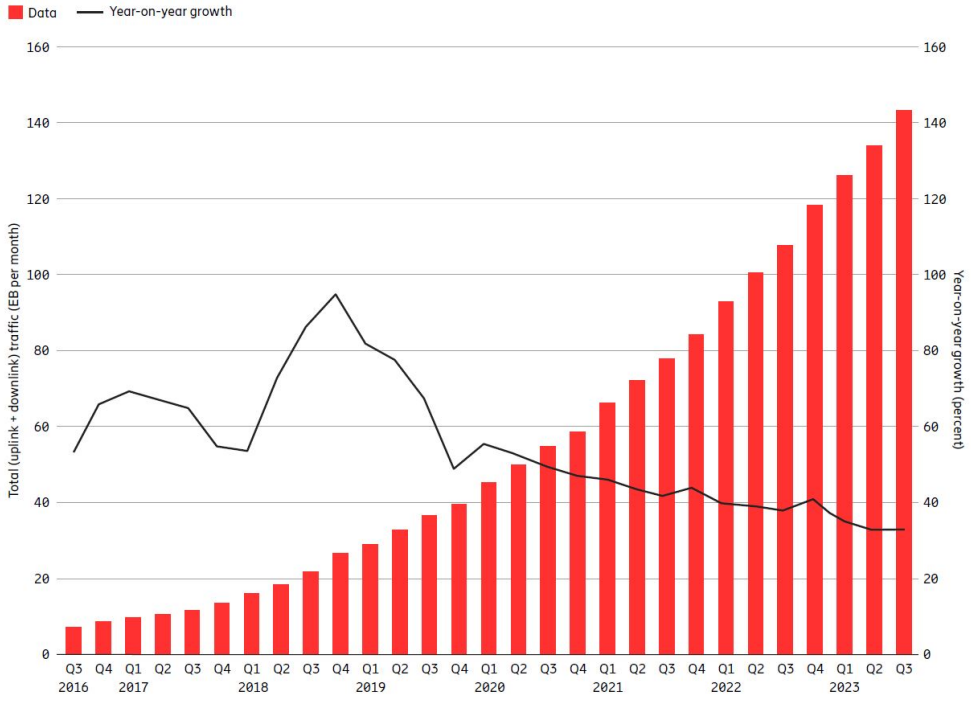

Figure 1.1: Global mobile network data traffic and year-on-year growth (EB per month) Ericsson (2021).

Traditional traffic classification methods, such as payload-based and port-based approaches, are becoming increasingly inadequate Zhao et al. (2021), Harish et al. (2020), Sun, Xue, Dong, Wang & Li (2010). The widespread adoption of encryption protocols like Hypertext Transfer Protocol Secure (HTTPS) Li et al. (2022) has rendered payload content opaque, as depicted in Figures 1.2 and 1.3. Similarly, the dynamic nature of port assignments complicates port-based classification Moore & Papagiannaki (2005). These limitations necessitate advanced techniques that can classify traffic accurately under encryption and obfuscation Nguyen & Armitage (2008).

Researchers are increasingly leveraging AI and ML techniques Yang et al. (2021) to overcome these challenges. This thesis focuses on addressing the crit-

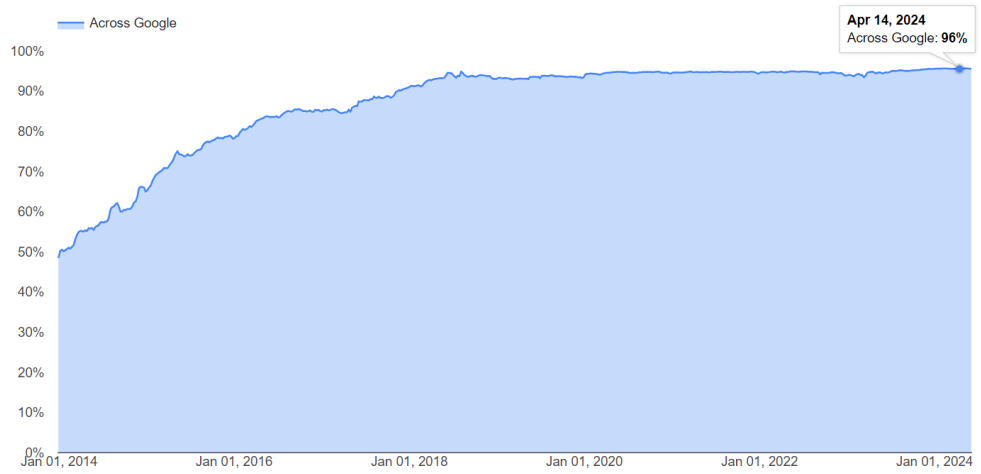

Figure 1.2: Percentage of Encrypted Traffic Across Google Services Google (2024).

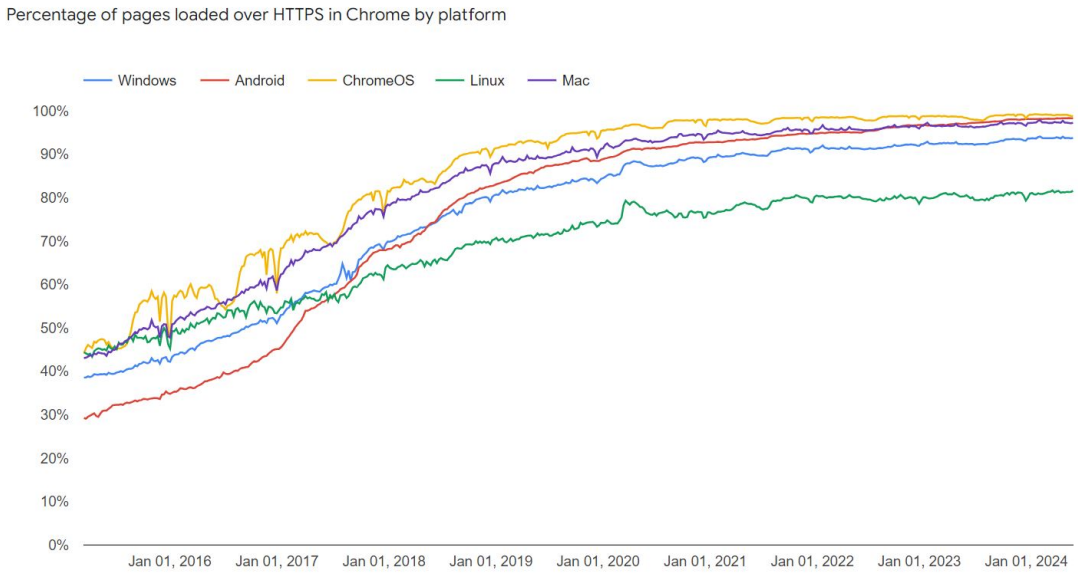

Figure 1.3: Percentage of Pages Loaded Over HTTPS in Chrome by Platform Google (2024).

ical need for identifying and prioritizing low-latency traffic in diverse network environments. Low-latency applications, such as video conferencing and online gaming, require minimal delay to function effectively, as shown in Figure 1.4. In contrast, non-low-latency applications, such as video streaming and file transfer, can tolerate higher delays Stockhammer (2011), Postel & Reynolds (1985).

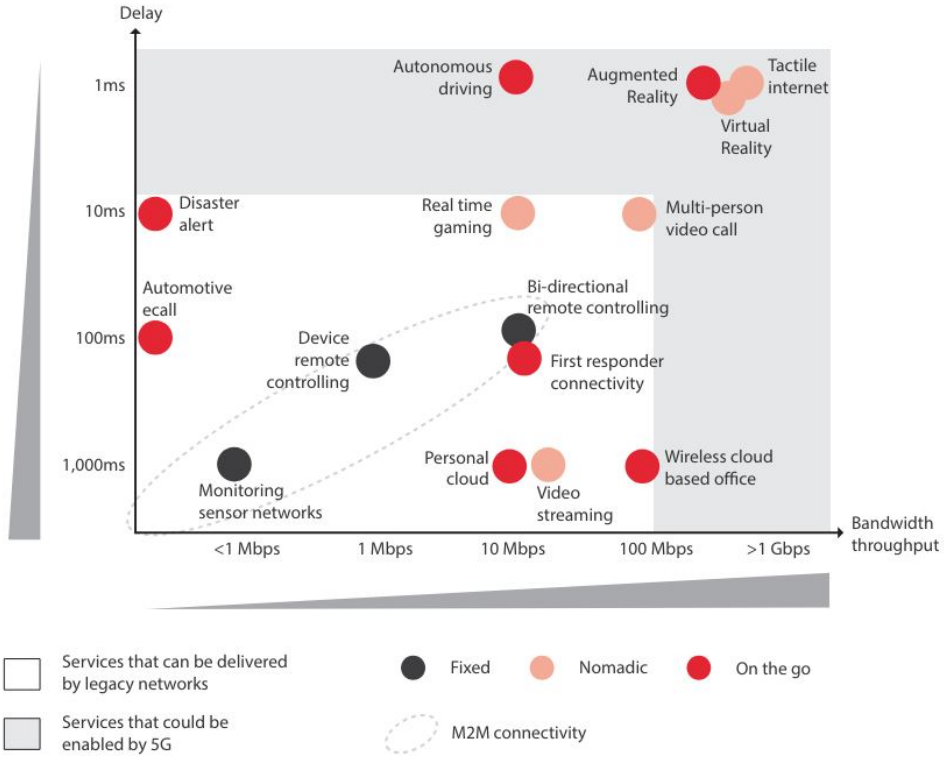

Figure 1.4: Delay and bandwidth requirements for various applications and services O'Connell et al. (2020).

This research introduces a novel approach that integrates Wavelet Transform and AI models to classify traffic efficiently. By capturing both time and frequency characteristics Chen & Liu (2014), the proposed method ensures accurate classification even under complex scenarios. Figure 1.5 highlights how Wavelet Transform transforms traffic data to reveal time-frequency patterns. These advance-

ments have significant implications for next-generation network management systems, ensuring better Quality of Service (QoS) for latency-sensitive applications.

Figure 1.5: Downlink Traffic Data Over Time and Wavelet Transform Analysis.

The increasing complexity of network traffic, coupled with the limitations of traditional classification methods, highlights the need for robust, adaptive solutions. By leveraging Wavelet Transforms and artificial intelligence, this research seeks to address these challenges, ensuring accurate classification of low-latency traffic. The next section outlines the specific research objectives that guide this study, reflecting the motivation discussed here.

1.2 Research Objectives

Building on the challenges identified in the **Background and Motivation**, this research aims to develop innovative methods for Internet traffic classification with a particular focus on low-latency traffic. The following objectives guide this research:

- **Trend-Based Feature Extraction**
  - **Objective:** Design algorithms to extract trend-based features that represent underlying traffic patterns.
  - **Rationale:** These features distinguish between low-latency and non-low-latency traffic based on periodicity and long-term trends Stockhammer (2011).
- **Integration with Wavelet Transform**
  - **Objective:** Combine WT techniques with trend-based features to analyze traffic characteristics in both time and frequency domains.
  - **Rationale:** WT captures transient features and localized variations, enhancing classification accuracy Yuan et al. (2010).
- **AI Model Training**
  - **Objective:** Train AI models using the extracted features to classify traffic accurately.
  - **Rationale:** Comprehensive feature sets improve the ability of AI models to generalize and recognize complex patterns Chen & Liu (2014).

- **Performance Evaluation**
  - **Objective:** Evaluate classification methods using metrics such as accuracy, precision, and recall.
  - **Rationale:** Extensive validation ensures the reliability and robustness of the proposed methods.

1.3 Contributions of the Thesis

This research makes several key contributions:

- **Novel Feature Extraction Techniques:** Introduces trend-based and WT-based features to enhance classification accuracy.
- **AI Model Integration:** Demonstrates significant improvements by integrating these features into AI models, such as ANN and SVM.
- **Extensive Evaluation:** Validates the proposed methods through rigorous experiments, achieving superior performance compared to traditional approaches.
- **Publications:** The findings have been disseminated through two peer-reviewed publications, highlighting the practical and theoretical advancements made.

1.4 Structure of the Thesis

This section provides a detailed road map to guide the reader through the thesis, emphasizing how each chapter contributes to the research objectives and builds

on the preceding sections.

- **Chapter 1: Introduction.** This chapter introduces the context, background, and motivation for the research. It defines the research objectives, highlights the contributions, and offers an overview of the thesis structure. It sets the stage for the discussions in the following chapters by outlining the problem statement and its significance in the field of network traffic classification.

- **Chapter 2: Theoretical Background.** This chapter provides the foundational knowledge necessary for understanding the proposed methods. It includes detailed discussions on Wavelet Transforms, feature extraction techniques, and artificial intelligence fundamentals. The theoretical background establishes the mathematical and conceptual tools required for the subsequent chapters, connecting these concepts to the research objectives.

- **Chapter 3: Related Work.** This chapter reviews the existing literature on network traffic classification, with a particular focus on traditional methods and their limitations. It also surveys modern approaches, such as machine learning and Wavelet Transforms, and identifies gaps that the thesis aims to address. The chapter positions the research within the broader context of the field, highlighting its novelty and relevance.

- **Chapter 4: Trend-Based Feature Extraction and Machine Learning Integration.** This chapter presents the development of trend-based feature extraction techniques and their integration with machine learning models. It details the design, implementation, and evaluation of these methods,

demonstrating their effectiveness in improving traffic classification accuracy. The chapter also connects these findings to the broader objectives of the thesis.

- **Chapter 5: Wavelet Transform and Deep Learning Integration.** This chapter extends the work presented in Chapter 4 by incorporating wavelet-based features into advanced AI models, such as artificial neural networks. It outlines the methodology, experimental setup, and results, showing how Wavelet Transforms enhance classification performance. The findings are critically analyzed, emphasizing their implications for real-world applications.

- **Chapter 6: Conclusion and Future Work.** The final chapter summarizes the thesis's key contributions and findings. It reflects on the research objectives and their achievement, discusses the implications of the results, and proposes directions for future research. The chapter provides a cohesive closure, tying together the insights gained from the entire study.

By following this structure, the thesis ensures a logical progression of ideas, enabling the reader to understand how each component contributes to the overall research goals.

CHAPTER 2

Theoretical Background

In this chapter, the essential theoretical concepts that underpin the contributions made in this thesis are explored. It begins by introducing the fundamentals of WT, including its role in time-frequency analysis and multiresolution signal decomposition. Following this, the chapter delves into feature extraction methods, focusing on statistical and trend-based techniques, which are crucial for capturing network traffic characteristics. The subsequent sections address the principles of network traffic analysis, including data packets, protocols, and different traffic types. The chapter concludes by exploring the foundations of artificial intelligence, particularly machine learning and deep learning models, which are integral to enhancing internet traffic classification accuracy.

2.1 Wavelet Transform Fundamentals

The Wavelet Transform has been developed as a potent mathematical tool for signal processing and analysis Rioul & Vetterli (1991). Unlike the Fourier Trans-

from, which represents signals as sums of sinusoids, the Wavelet Transform describes signals as a combination of functions localized in both time and frequency. This property makes wavelets especially useful for examining non-stationary signals with frequency content that changes over time. The Wavelet Transform relies on the idea of wavelets, small waves with finite duration and an average value of zero. These wavelets are used to decompose a signal into components at different scales and positions, facilitating multiresolution analysis.

The fundamental concepts of Wavelet Transform include:

- **Wavelets and Scaling Functions:** A wavelet is a function generated by translating and dilating a prototype wavelet, also known as the mother wavelet, denoted as $\psi(t)$. Scaling functions provide a complementary role by defining approximations at coarser scales. These functions are critical in constructing multi-resolution analysis, as evident in the hierarchical nature of the "Wavelet Transform" section in Figure 2.1, where different scales (y-axis) reflect frequency-related components at varying time resolutions (x-axis).

- **Multi Resolution Analysis (MRA):** MRA decomposes a signal into components of varying resolutions or scales, offering a hierarchical representation of the signal. In Figure 2.1, the "Wavelet Transform" plot demonstrates this capability through its ability to represent both high-frequency details at finer scales and low-frequency trends at coarser scales, captured by localized regions in time and scale.

- **Time-Frequency Localization:** Unlike the Fourier Transfrom, which pro-

vides only frequency-domain information, and Short Time Fourier Transform (STFT), which offers a limited fixed-window time-frequency representation, Wavelet Transform adapts its resolution dynamically. This adaptability, visualized in Figure 2.1, is achieved by using shorter windows for high frequencies and longer windows for low frequencies, ensuring optimal localization of transient features in time and frequency domains.

- **Flexibility of Representation:** Wavelet Transform utilizes a scalable approach, where the signal is analyzed at various levels of detail. This flexibility is demonstrated in Figure 2.1, as the "Wavelet Transform" graph smoothly transitions from high-resolution details to broader trends, unlike the fixed window size of STFT that results in uniform resolution.

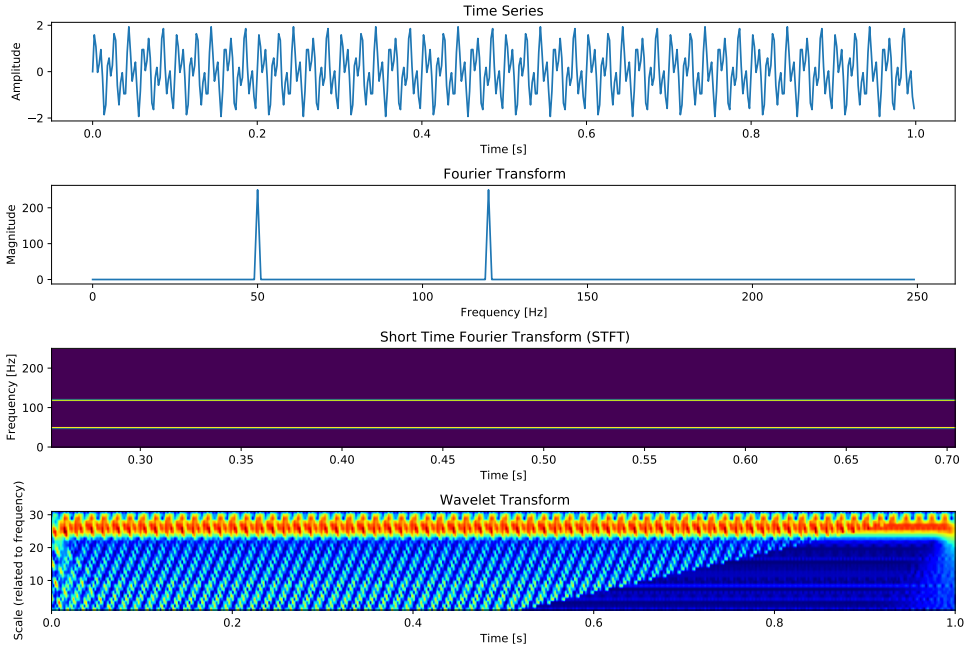

Figure 2.1: The time and frequency resolutions of different methods (python generated).

Figure 2.1 provides a comparative view of various transforms. The "Fourier Transfrom (FT)" demonstrates the strength of FT in capturing frequency content, albeit without time localization. The "Short Time Fourier Transform" addresses this limitation by using fixed-size windows for a time-frequency tradeoff. However, the "Wavelet Transform" surpasses STFT by employing scale-adaptive windowing, yielding more precise localization in both time and frequency.

The mathematical framework of Wavelet Transform is rooted in functional analysis and involves several key components:

Continuous Wavelet Transform (CWT) of a signal $f(t)$ is defined as the convolution of $f(t)$ with a scaled and translated version of the mother wavelet $\psi(t)$. Mathematically, it is expressed as:

$$W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\psi^* \left(\frac{t-b}{a} \right) dt \tag{2.1}$$

where a and b are the scaling and translation parameters, respectively, and $\psi^*(t)$ denotes the complex conjugate of $\psi(t)$.

Discrete Wavelet Transform (DWT) is a sampled version of the CWT, where the scales and positions are seperated. The DWT is particularly useful for digital signal processing and is computed using filter banks. The approximation coefficients A_j and detail coefficients D_j are obtained through the iterative application of low-pass and high-pass filters followed by down-sampling.

Wavelet Basis Functions , the Wavelet Transform relies on a set of orthogonal or biorthogonal basis functions. These basis functions are derived from the mother wavelet and scaling function through dilation and translation. The or-

thogonality property ensures that the decomposition is reversible, allowing for perfect reconstruction of the original signal.

Various types of wavelets have been developed for different applications and signal characteristics. Some of the prominent wavelets include:

- **Haar Wavelet:** is the simplest and oldest wavelet, introduced by Alfred Haar in 1910 Sun, Jiang & Tian (2010). It is a piecewise constant function that provides a basic form of Wavelet Transform. The Haar scaling function $\phi(t)$ and wavelet function $\psi(t)$ are defined as:

$$\phi(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$\psi(t) = \begin{cases} 1 & \text{if } 0 \leq t < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

- **Daubechies Wavelets:** Named after Ingrid Daubechies, these wavelets are characterized by their compact support and smoothness. The Daubechies wavelets, denoted as D_N, have scaling functions and wavelet functions that are solutions to specific recursive relations. The Daubechies-4 wavelet (D_4) has the scaling coefficients h_k given by:

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}} \tag{2.4}$$

- **Coiflets:** introduced by Ronald Coifman, are designed to have both the wavelet function and scaling function with vanishing moments. This prop-

erty makes them suitable for signal approximation and denoising tasks. The Coiflet wavelet $\psi(t)$ and scaling function $\phi(t)$ satisfy the following vanishing moments conditions:

$$\int t^k \phi(t) dt = 0, \quad \int t^k \psi(t) dt = 0, \quad k = 0, 1, \ldots, N-1 \tag{2.5}$$

- **Symlets:** Symlets wavelets, are a modified version of Daubechies wavelets with improved symmetry. They provide better phase response and are often used in biomedical signal processing. The construction of Symlets involves finding a set of filter coefficients that satisfy specific symmetry conditions.

- **Meyer Wavelet:** The Meyer wavelet is an infinitely differentiable wavelet with compact support in the frequency domain. It is particularly useful for applications requiring smooth wavelet functions, such as audio signal processing. The Meyer wavelet $\psi(t)$ is defined in the frequency domain as:

$$\hat{\psi}(\omega) = \begin{cases} \sin\left(\frac{\pi}{2}\nu\left(\frac{3|\omega|}{2\pi} - 1\right)\right) & \text{if } \frac{2\pi}{3} \leq |\omega| \leq \frac{4\pi}{3} \\ \cos\left(\frac{\pi}{2}\nu\left(\frac{3|\omega|}{4\pi} - 1\right)\right) & \text{if } \frac{4\pi}{3} \leq |\omega| \leq 2\pi \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where ν is a smooth transition function.

In this thesis, the Wavelet Transform is applied to capture both the time and frequency characteristics of network traffic, making it particularly suitable for analyzing dynamic and non-stationary traffic patterns. By decomposing traffic data into multiple scales, the Wavelet Transform allows for

the identification of transient features and localized variations, which are crucial for accurately classifying low-latency traffic. This method, when combined with AI models, enhances the precision and adaptability of the classification system, offering a robust solution for modern, heterogeneous network environments.

2.2 Feature Extraction

Feature Extraction is the process of deriving significant attributes from raw data, capturing essential characteristics and patterns. These features become very important for most analytical and predictive modeling operations; especially, it is of great help in analyzing data associated with different time series. This section discusses basic statistical measures and techniques for extracting these features to provide a theoretical basis for subsequent application.

2.2.1 Statistical Features

Statistical features are essential for capturing key characteristics of time-series data. For instance, Figure 2.2 illustrates the concept of mean, showing how it represents the central tendency of the data over time. Similarly, Figure 2.3 highlights variance as a measure of the data's spread, with standard deviation bands depicted to emphasize variability around the mean. These features are foundational in distinguishing traffic patterns. The primary statistical measures include:

- **Mean:** The mean or average is a measure of central tendency, representing the typical value in the data set. It is calculated as:

$$\mu = \frac{1}{N} \sum_{t=1}^{N} X_t \tag{2.7}$$

where X_t represents the data points and N is the number of observations. The mean provides a summary measure indicating the central point of the data distribution. Figure 2.2 is the demonstration of the time series data with the mean value indicated by a red dashed line.

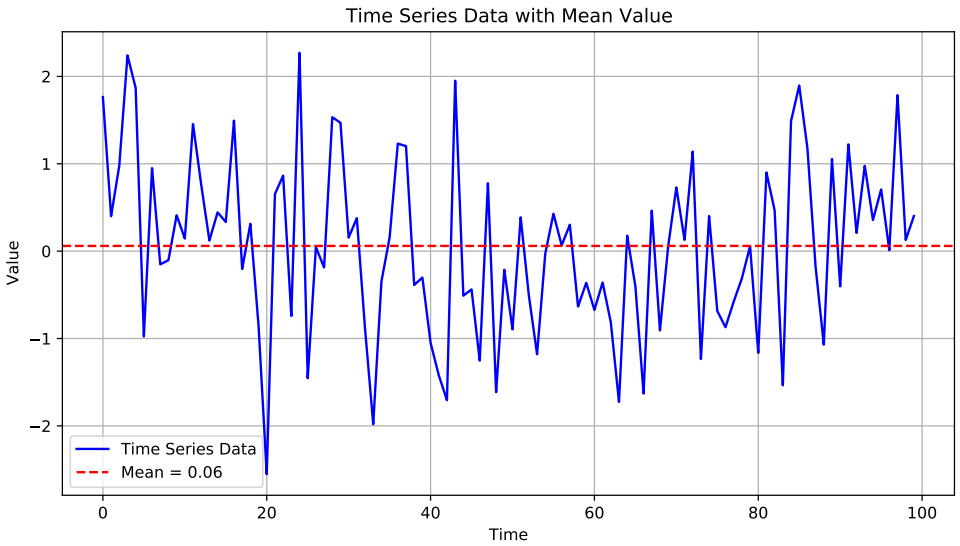

Figure 2.2: Time series data with the mean value.

- **Variance:** Variance measures the dispersion or spread of the data around the mean. It is given by:

$$\sigma^2 = \frac{1}{N} \sum_{t=1}^{N} (X_t - \mu)^2 \tag{2.8}$$

where μ is the mean of the data. A higher variance indicates more spread out data points, while a lower variance indicates that data points are closer

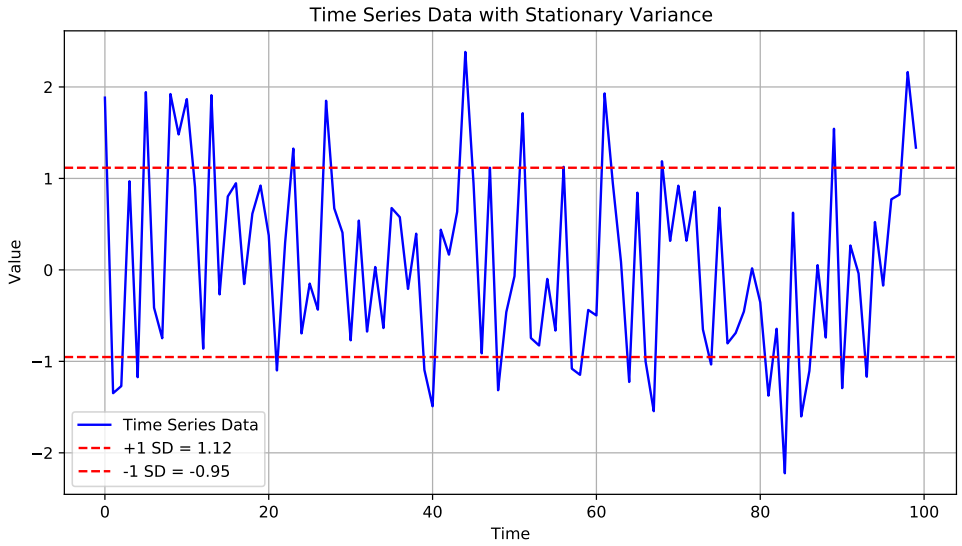

Figure 2.3: Time series data with stationary variance.

to the mean. Figure 2.3 is the plot of the time series data with stationary variance.

- **Skewness:** Skewness indicates the asymmetry of the data distribution around the mean. It is calculated as:

$$\text{Skewness} = \frac{1}{N}\sum_{t=1}^{N}\left(\frac{X_t - \mu}{\sigma}\right)^3 \tag{2.9}$$

where σ is the standard deviation. Positive skewness indicates a distribution with a long right tail, while negative skewness indicates a distribution with a long left tail.

- **Kurtosis:** Kurtosis measures the 'tailedness' of the data distribution. It is given by:

$$\text{Kurtosis} = \frac{1}{N}\sum_{t=1}^{N}\left(\frac{X_t - \mu}{\sigma}\right)^4 - 3 \tag{2.10}$$

The subtraction of 3 ensures that the kurtosis of a normal distribution is zero. Higher kurtosis indicates more extreme outliers.

2.2.2 Statistical Feature Extraction Techniques

Several techniques are used to extract meaningful statistical features from raw time-series data. These techniques help identify patterns, trends, and relationships within the data.

- **Autocorrelation Function (ACF):** In time series analysis, the covariance function at lag k is often called the Autocorrelation Function at lag k because it represents the covariance between Z_t and $Z_t + k$ from the same process, separated by k time lags Little (2013). Hence, the ACF between Z_t and $Z_t + k$ is simply the standardized autocovariance function,

$$\rho_k = \frac{\sum_{t=1}^{N-k}(Z_t - \mu)(Z_{t+k} - \mu)}{\sum_{t=1}^{N}(Z_t - \mu)^2} \tag{2.11}$$

where we note that for a stationary process, $\text{Var}(Z_t) = \text{Var}(Z_{t+k}) = \gamma_0$. It is easy to see that for a stationary process, the ACF has the properties:

1. $\rho_0 = 1$
2. $|\rho_k| \leq 1$
3. $\rho_{-k} = \rho_k$, which follows from the fact that the time difference between Z_t and Z_{t+k} and between Z_t and Z_{t-k} are the same; and

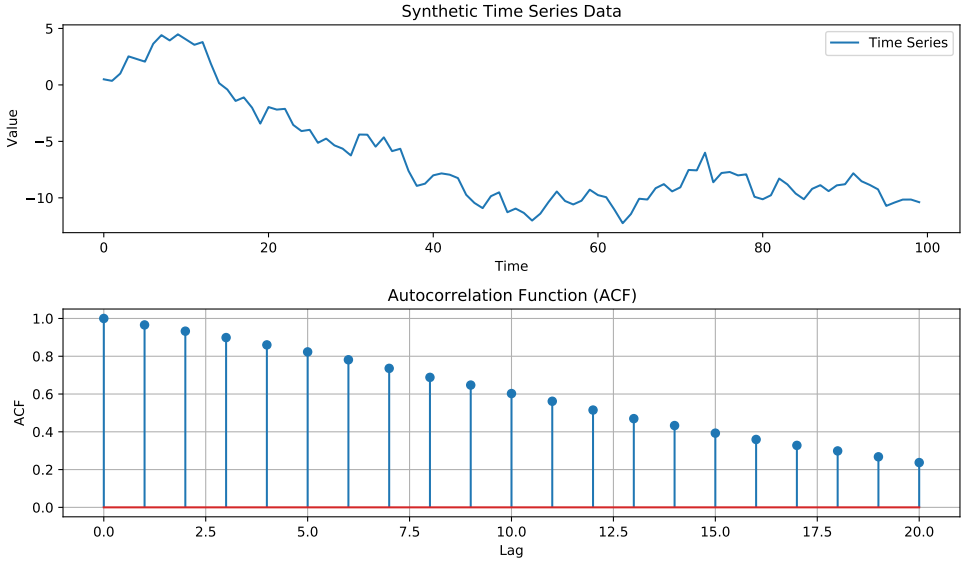

Figure 2.4: Synthetic time series data and ACF.

4. ρ_k is positive semidefinite – that is,

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\rho_{|t_i-t_j|} \geq 0,$$

for any set of time-points, $t_1, \ldots, t_n$ and any real numbers $\alpha_1, \ldots, \alpha_n$.

Using property 3, we plot an ACF only for the non-negative lags, which is also called a correlogram. Figure 2.4 demonstrates the synthetic time series data with its ACF.

- **Partial Autocorrelation Function (PACF):** The PACF measures the correlation between observations of a time series separated by different lags, controlling for the correlations at shorter lags. It is given by:

$$\phi_{kk} = \text{Corr}(X_t, X_{t+k}|X_{t+1}, \ldots, X_{t+k-1}) \tag{2.12}$$

The PACF is useful for determining the order of autoregressive models.

Unlike the ACF, the PACF of a stationary AR(p) process cuts off after lag p, indicating no further partial correlation beyond this point.

- **Moving Average (MA)**: A moving average is a widely used statistical technique to smooth time series data by defining an average of different subsets of the complete data set. Reducing noise and short-term fluctuations from data is the key focus of the moving average. It has a special place in the analysis of financial data for forecasting stock prices and economic time series. The Simple Moving Average (SMA) is the most basic form of moving average. It calculates the unweighted mean of the previous N data points. The formula for the simple moving average at time t with a window size N is:

$$\text{SMA}_t = \frac{1}{N} \sum_{i=0}^{N-1} X_{t-i} \tag{2.13}$$

where X_{t-i} represents the data points at times $t, t-1, \ldots, t-(N-1)$. The SMA is effective for smoothing data, but it assigns equal weight to all observations within the window, which can be a limitation when the most recent observations are more relevant. Figure 2.5 demonstrates the application of a moving average to time-series data, smoothing out short-term fluctuations to reveal long-term trends. The red line, representing a 14-period MA, overlays the raw time series to emphasize overall directional movement. This technique is vital in reducing noise and identifying consistent patterns in network traffic.

- **Exponential Moving Average (EMA)**: The EMA addresses the limitation of the SMA by assigning more weight to recent observations. The EMA for

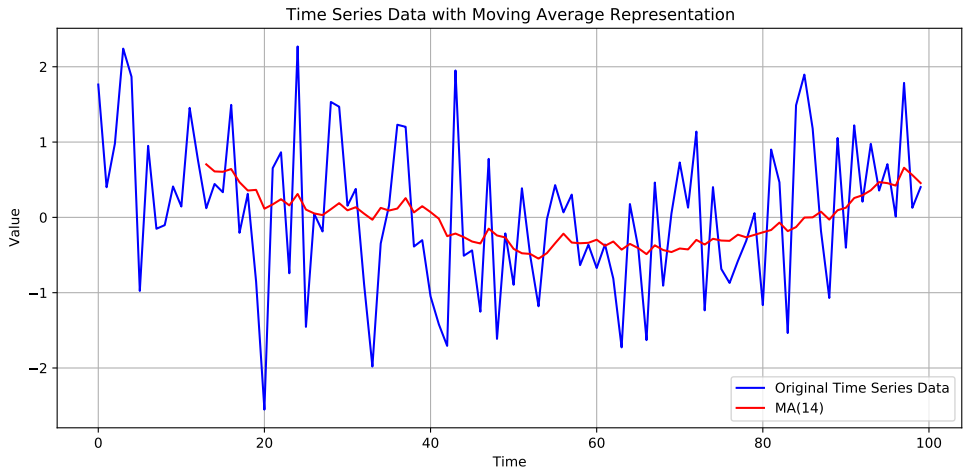

Figure 2.5: Time series data with the moving average.

a given time t is calculated using the following recursive formula:

$$\text{EMA}_t = \alpha X_t + (1 - \alpha)\text{EMA}_{t-1} \tag{2.14}$$

where $0 < \alpha < 1$ is the smoothing factor, which determines the weight given to the most recent observation X_t. The value of α is usually chosen as $\alpha = \frac{2}{N+1}$, where N is the window size. The EMA responds more quickly to recent changes than the SMA, making it more suitable for time series with significant short-term variations.

- **Weighted Moving Average (WMA):** The WMA assigns different weights to different observations, typically giving more importance to recent data points. The formula for the WMA at time t with window size N and weights w_i is:

$$\text{WMA}_t = \frac{\sum_{i=0}^{N-1} w_i X_{t-i}}{\sum_{i=0}^{N-1} w_i} \tag{2.15}$$

where w_i are the weights assigned to the observations X_{t-i}. The weights can be chosen based on the specific requirements of the analysis, ensuring that more recent observations have a greater influence on the moving average.

- **Autoregressive (AR):** AR models represent the value of a time series as a linear combination of its past values. An AR model of order p is defined as:

$$X_t = \mu + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \epsilon_t \tag{2.16}$$

where ϕ are the parameters and ϵ_t is the error term. AR models are useful for modeling long-term dependencies in the data.

- **Ratio:** The ratio is used to compare two different quantities, providing insights into the relative size of one quantity with respect to another. For example, the ratio of two consecutive data points X_t and X_{t-1} is given by:

$$\text{Ratio} = \frac{X_t}{X_{t-1}} \tag{2.17}$$

This feature helps to identify relative changes in the data. Ratios are particularly useful in financial time series and other domains where relative changes are more informative than absolute changes.

- **Slope:** The slope measures the rate of change in the data over time. It is calculated as the difference between consecutive data points, normalized by the time interval. For a time series X_t, the slope is given by:

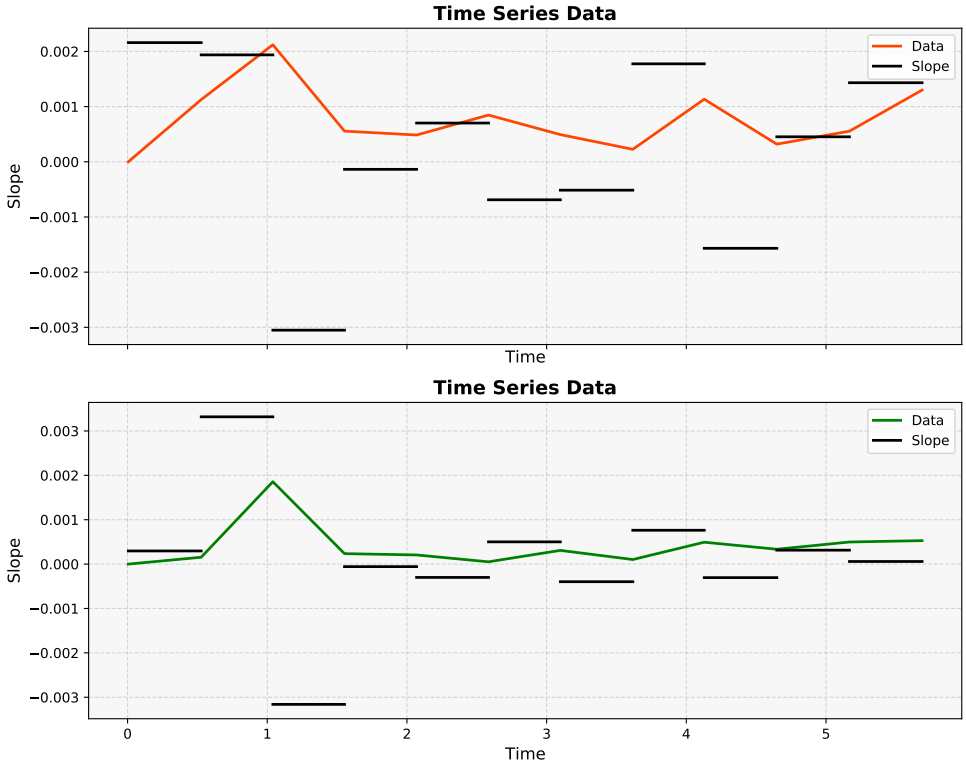

Figure 2.6: Time series data with the Slope.

$$\text{Slope} = \frac{X_t - X_{t-1}}{t - (t-1)} \tag{2.18}$$

The slope provides insight into the direction and magnitude of the changes in the data. It is useful for identifying trends and rates of change in the data over time. The slope, illustrated in Figure 2.6, captures the rate of change in time-series data, offering a dynamic view of trends. The figure highlights two datasets, where the calculated slopes provide immediate insights into increasing or decreasing patterns. This feature is particularly useful in identifying bursts or steady flows in network traffic, which are critical for distinguishing between latency-sensitive and tolerant traffic

types."

2.3 Network Traffic Analysis

Network traffic analysis plays a vital role in managing, securing, and optimizing the performance of a network Mala et al. (2023). This discipline undertakes a detailed examination of the data packets under transmission to appreciate, monitor, and ensure the network infrastructure. Network traffic analysis helps the administrator to identify usage patterns, detect anomalies, and enforce adequate security measures Latif et al. (2022). This section elaborates on critical theoretical concepts, methodologies, and challenges surrounding network traffic analysis. The knowledge of network traffic is also quite significant in keeping modern networks healthy and highly efficient. As the size and complexity of networks grow, the ability to monitor and analyze traffic is also important. Traffic analysis helps identify where a bottleneck is located, ensures policy compliance, detects attempted intrusions, and manages bandwidth.

Network traffic analysis can broadly be classified into two parts: real-time analysis, which is monitoring live traffic to detect issues as and when they take place, and offline analysis, which is the analysis of past traffic data to detect trends and patterns. Real-time is an essential function for immediate threat detection and response, while offline analysis gives one a peek into long-term network performance and usage trends. A typical network traffic analysis process includes; **Data Collection**, **Data Preprocessing** and **Data Analysis**. Figure 2.7 illustrates a basic example of network traffic volume over time, highlighting peak usage periods and potential bottlenecks. By systematically analyzing net-

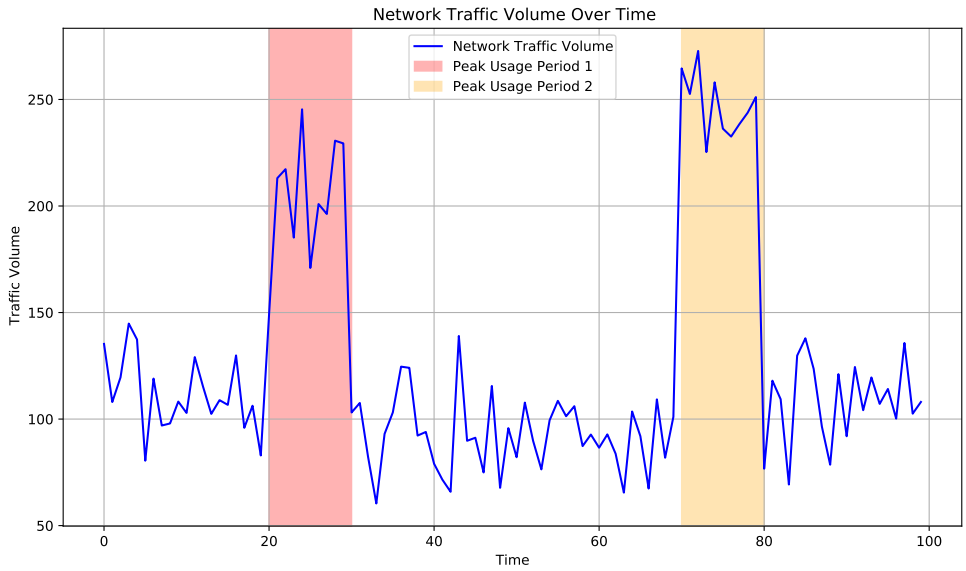

Figure 2.7: Example graph showing network traffic volume in KB over time. Peak usage periods are highlighted, indicating times when the network is most heavily utilized.

work traffic, organizations can enhance their network's performance, improve security posture, and ensure efficient utilization of resources.

2.3.1 Data Packets

Data packets are the basic units for data transmission over a computer network. They encapsulate the data sent from one device to another through a network. Proper knowledge of the structure and functioning of data packets is essential in conducting a practical network traffic analysis so network administrators can properly monitor, diagnose, and secure communications within their networks. A typical data packet consists of three main components: the header, the payload, and the trailer (if present). Each component serves a specific purpose in

ensuring that data is transmitted accurately and efficiently.

- **Header:** The header contains control information required for routing and delivery of the packet. The Figure 2.8 shows a visual representation of fields within the IPv6 header, detailing their size and role within the IPv6 packet structure.

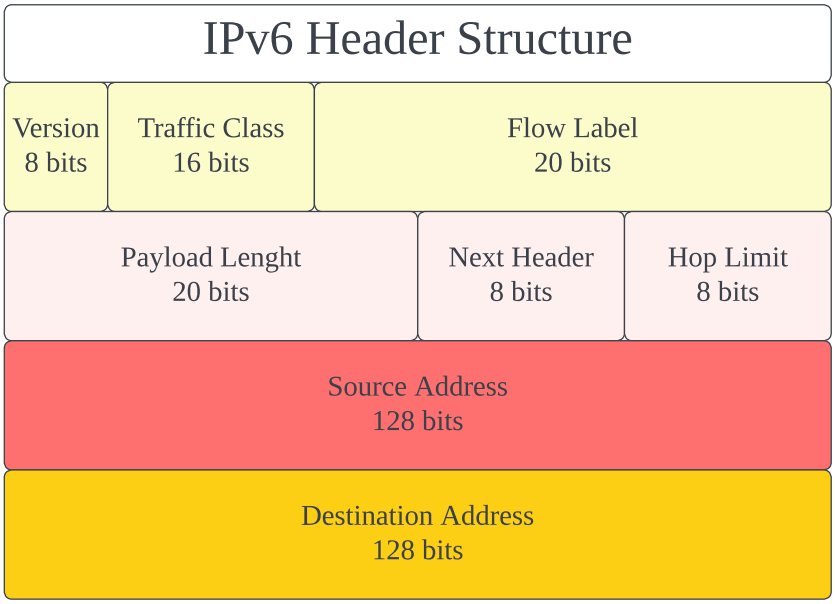

Figure 2.8: Internet Protocol version 6 (IPv6) Header structure format and fields.

  - **Source Address:** The IP address of the device that sent the packet.
  - **Destination Address:** The IP address of the device intended to receive the packet.
  - **Version:** The size of this field is 4-bit. Indicates the version of the

Internet Protocol, which is always 6 for IPv6, so the bit sequence is 0110.

- **Traffic Class:** The Traffic Class field indicates class or priority of IPv6 packet which is similar to Service Field in IPv4 packet. It helps routers to handle the traffic based on the priority of the packet.
- **Flow Label:** Flow Label field is used by a source to label the packets belonging to the same flow in order to request special handling by intermediate IPv6 routers, such as non-default quality-of-service or real-time service.
- **Payload Length:** It is a 16-bit field, indicates the total size of the payload which tells routers about the amount of information a particular packet contains in its payload.
- **Next Header:** Next Header indicates the type of extension header(if present) immediately following the IPv6 header.
- **Hop Limit:** Hop Limit field is the same as TTL in IPv4 packets. It indicates the maximum number of intermediate nodes IPv6 packet is allowed to travel.

- **Payload:** The payload is the actual data being transported. It contains the user data or application data that needs to be transmitted from the source to the destination. The payload size can vary depending on the network'sMaximum Transmission Unit (MTU).
- **Trailer:** The trailer (or footer) may include a Frame Check Sequence (FCS) or error-detection code used to verify the integrity of the transmitted data. It is more common in link-layer protocols (e.g., Ethernet frames).

2.3.2 Traffic Flow

Traffic flow in network communication refers to the sequence and behavior of data packets transmitted from a source to a destination over a network. Traffic flow can be characterized by several parameters, which provide insights into the nature and behavior of the network traffic:

- **Source and Destination:** This indicates the IP addresses and port numbers of the sender and receiver, hence defining the endpoints of the traffic flow. For instance, a flow from a Web server to a client would have, as its source, the server's IP plus port 80 (HTTP) and, as its destination, the client's IP along with a random high-numbered port.
- **Protocol:** The transport layer protocol (e.g., TCP, UDP) being used for the communication. Different protocols have different flow properties; for example, TCP flows have three connection phases set-up, maintenance, and tear-down, while the UDP protocol does not have connections since it is connectionless and delivery is not guaranteed.
- **Volume:** The total amount of data transferred during the flow, typically measured in bytes or packets. This parameter helps in understanding the bandwidth usage and load on the network.
- **Duration:** The time span over which the flow occurs, from the first packet to the last. Duration is crucial for analyzing the persistence and stability of network connections.
- **Packet Rate:** he number of packets transmitted per unit time, usually measured in Packets Per Second (PPS). High packet rates can indicate bursty

traffic patterns or potential denial-of-service attacks.

- **Interarrival Time:** The time interval between consecutive packets in a flow. Analyzing interarrival times helps in identifying traffic patterns and detecting anomalies such as jitter in real-time applications.

2.3.3 Network Traffic Types

	Live Streaming	VoIP	Video Traffic	HTTP Download	Video Conference
Real Time	YES	YES	NO	NO	YES
Delay Sensitivity	SENSITIVE	SENSITIVE	INSENSITIVE	INSENSITIVE	SENSITIVE
Drop Sensitivity	SENSITIVE	SENSITIVE	INSENSITIVE	INSENSITIVE	SENSITIVE
Benign/Greedy	BENIGN/GREEDY	BENIGN	BENIGN/GREEDY/	GREEDY	BENIGN/GREEDY
Smooth/Bursty	SMOOTH/BURSTY	SMOOTH	BURSTY	BURSTY	SMOOTH/BURSTY
Applications	TWITCH,YOUTUBE	VOIP	NETFLIX,YOUTUBE	FILE TRANSFER	ZOOM, SKYPE

Figure 2.9: Characteristic of different Internet traffic types.

Network traffic types refer to the different kinds of data transmitted over a network, each with its own characteristics, requirements, and behaviors. Figure 2.9 summarizes the characteristics of various network traffic types, including live streaming, file transfer, and VoIP. The figure visually categorizes traffic based on key attributes like latency sensitivity and burstiness. Such distinctions are instrumental for prioritizing traffic in network management, particularly for

ensuring quality of service in latency-critical applications. Understanding these traffic types is essential for effective network management, optimization, and security. Below, some of the common network traffic types are described:

Live Streaming

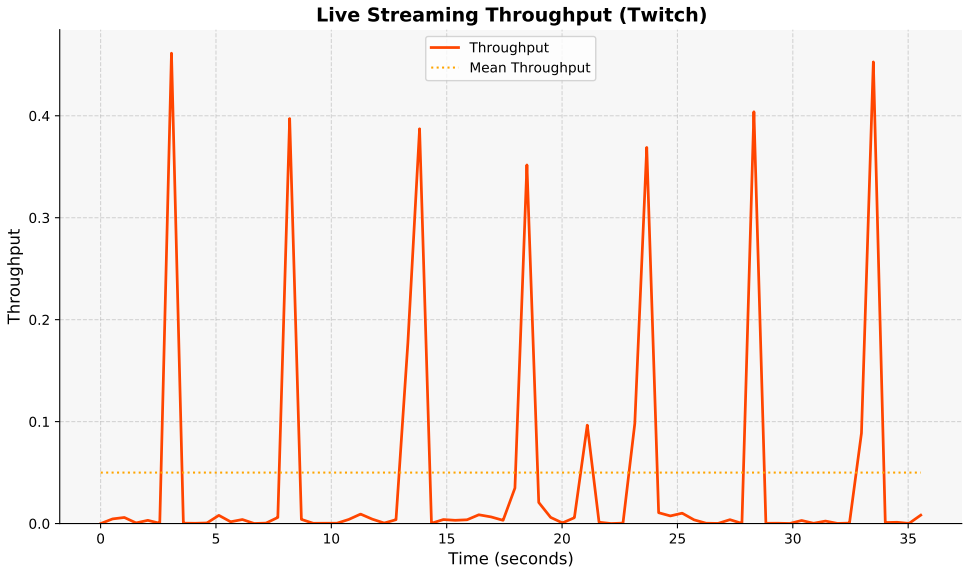

Figure 2.10: Overall nature of live streaming traffic.

Live streaming traffic represents the continuous transmission of audio and video content from a server to multiple clients in real time. This class of traffic requires both low-latency and high bandwidth to allow for smooth, buffering-free playback.

Popular live streaming protocols include the RTMP, HTTP Live Streaming (HLS), and Dynamic Adaptive Streaming over HTTP (DASH). Most of these application protocols are based on TCP for dependable data delivery, while for lower latency, some make use of UDP. Nature of the live streaming traffic demonstrated in Figure 2.10. This nature of traffic both greedy and benign for

bandwidth utilization, it buffers during the on state(greedy) and waits during the off state (benign).

Live-streaming traffic is bandwidth-sensitive and sensitive to latency. The adaptive bitrate streaming techniques change the quality of the stream to a level that best fits the client's available bandwidth.

Voice Over IP

VoIP traffic can communicate over IP networks by converting analog voice signals into IP data packets. This nature of the packet requires low-latency and minimum jitter to make uninterruptible, clear voice calling. VoIP usually uses Session Initiation Protocol (SIP) for call setup and management and RTP for audio data transmission. VoIP generally works over UDP to minimize delay.

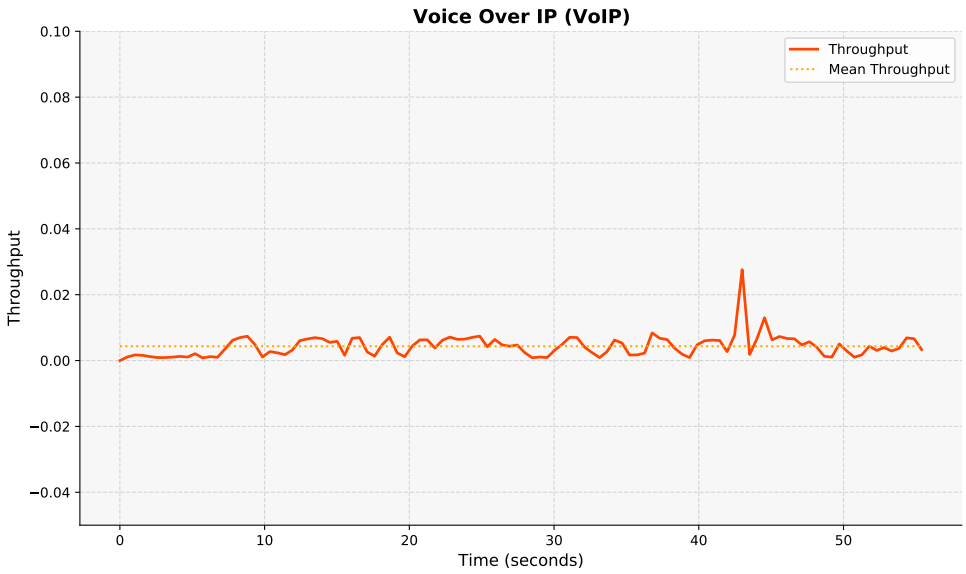

Figure 2.11: Overall nature of VoIP traffic.

VoIP traffic is sensitive to delay, jitter, and packet loss. So, VoIP packets are provided with the mechanisms from QoS that are placed on priority levels to

ensure quality in calls. Nature of the VoIP traffic demonstrated in Figure 2.11. This nature of traffic does not greedy for bandwidth utilization but seeks for constant packet transfer for clear call.

Video Traffic

Video traffic transfers pre-recorded video content from servers to clients over the Internet. This type of traffic is one of the most bandwidth-intensive, with unique requirements and characteristics to ensure smooth playback. For example, there are services such as Netflix and YouTube, which highly generate traffic from videos. On the other hand, video traffic becomes very consuming in cases of high-definition and ultra-high-definition content. Nature of this traffic is similar with live streaming with only one difference which is video traffic does not latency sensitive.

Video data rates for streaming video can vary significantly based on both resolution and the techniques used in compressing. For example, a 1080p HD stream takes about 5 Mbps, while 4K UHD takes 25 Mbps or more. These video streaming services implement buffering and caching, such that the player buffers a part of the video to be played even before the actual video playback begins. It is held in a buffer to make available smooth video playback without interruptions. This buffer can do an equalizer for network variability and temporary data transmission interruptions. CDNs also reduce latency and improve delivery speed through content caching closer to the user.

HTTP Downloads

Download traffic is a file transfer from a web server to a client using HTTP or HTTPS. The downloads may take the form of a small file, such as an image or document, or big ones, such as software or multimedia. A download takes place over TCP, and HTTPS will add an encryption layer to make it secure. Big file downloads might use chunked transfer encoding with the resumption of interrupted downloads.

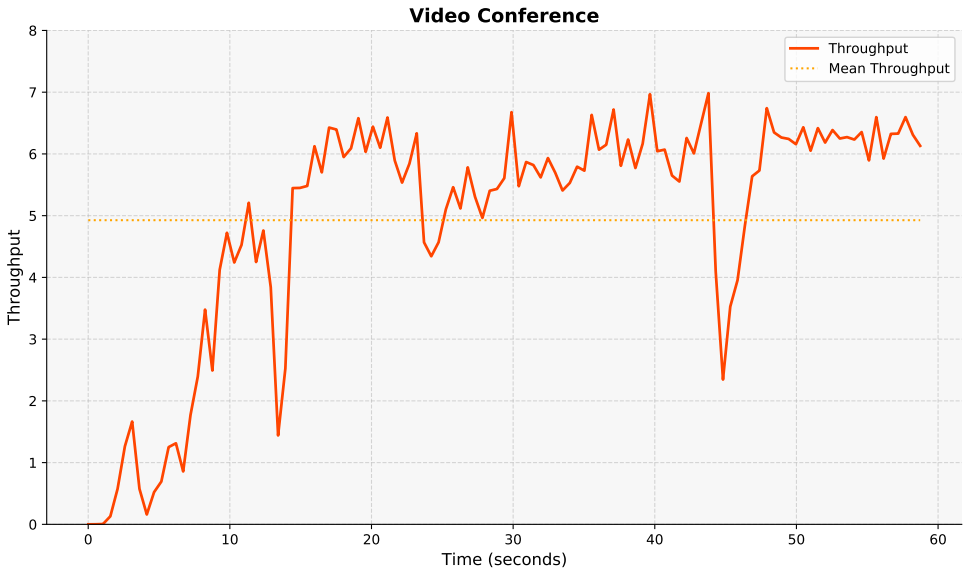

Figure 2.12: Overall nature of HTTP download traffic.

Downloading traffic is often bandwidth-intensive, especially in the case of large files. As seen in Figure 2.12, starting from 0 to max available bandwidth utilization represents the nature of this traffic. It also can be bursty, meaning there are periods of high activity followed by periods of inactivity.

Torrents

Traffic on torrents uses the BitTorrent protocol to share files using Peer-to-Peer (P2P) technology. A file is downloaded from many peers instead of just one server source to a client; hence, it makes the downloading process efficient and fast. It is both TCP and UDP driven and is basically run over a tracker that coordinates peer discovery and, in general DHT, based. How tit-for-tat pieces of the file are exchanged is where peers exchange pieces of the file. Torrent traffic can be bandwidth intensive and produces high upload and download traffic. It is fault-tolerant to network failures due to its distributed nature.

2.3.4 Flow Analysis

Flow analysis is a basic network traffic analysis technique that involves collecting and analyzing packet sequences from the source to the destination. It tries to understand the traffic patterns, raise alerts of the anomalies, and optimize network performance by analyzing flows instead of just packets. Network flow is defined about the source and destination IP addresses, ports, and the protocol used. A flow is defined as a sequence of packets with shared characteristics, often expressed through a tuple that includes the source IP address, the destination IP address, the source port, the destination port, and the protocol. An abstract representation of all packets of the corresponding flow in a flow record contains statistics like the number of packets, number of bytes, start time, end time, and flags for the TCP protocol.

Flow data is collected using flow exporters on routers or switches, which aggregate packets into flows and periodically export flow records to a flow collec-

tor. Standard flow export protocols include NetFlow Cisco (2024), sFlow sFlow (2024), and Internet Protocol Flow Information Export (IPFIX) IPFIX (2024). A flow collector stores flow records in a database and thus allows one to carry out analysis tasks like identifying top talkers, detecting traffic patterns that are not common, extracting the distribution of the traffic by application or protocol, and even capacity planning.

Several metrics are calculated from flow data to analyze network traffic:

$$F(FlowRate) = \frac{\text{Number of Flows}}{\text{Observation Period}} \tag{2.19}$$

$$P(PacketRate) = \frac{\text{Number of Packets}}{\text{Duration of Flow}} \tag{2.20}$$

$$B(ByteRate) = \frac{\text{Number of Bytes}}{\text{Duration of Flow}} \tag{2.21}$$

Flow analysis also involves anomaly detection by comparing current flow metrics to historical baselines using statistical methods like mean and standard deviation:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{2.22}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2} \tag{2.23}$$

An anomaly might be detected if the current value X_t of a metric falls outside the range $[\mu - 3\sigma, \mu + 3\sigma]$.

2.4 Artificial Intelligence Fundamentals

The concept of Artificial Intelligence (AI) is well established among the scientific, technical, engineering, and mathematical communities, but it recently became widely known in the popular media, where it is often used in a loose way to refer to somewhat abstract concepts. Although it is the most human-centered approach to the definition of AI, historically, several have been followed with the idea to create machines that will perform as many tasks as people can do. In the ideal case, it seems even more effective than humans. The quintessential idea of this approach is the famous Turing Test Turing (2009), designed to provide an objective threshold for establishing machine intelligence. Currently, the discipline has grown and divided so much that we can state definite vital topics that would allow the computer to pass off as intelligent Russell & Norvig (2016) are natural language processing for communication, knowledge representation to store what it knows or learns, automated reasoning relying on data to answer questions and present new information, Machine Learning (ML) to adapt to novel situations and to recognise and extrapolate patterns.

2.4.1 Machine Learning

Machine Learning, a subfield of Artificial Intelligence, automatically processes data. With voluminous datasets being available in modern days, it is pertinent for Machine Learning techniques to adapt and maximize the potentials in the datasets.

Generally, ML can be categorised into two central types: supervised and unsupervised learning Murphy (2012). The objective of a supervised kind of

learning is to learn the relationship or map from the inputs to the outputs given labeled pairs. On the other hand, unsupervised learning does not have any labeled outputs to compare the input data; it mainly recognizes patterns in the input data. Such a learning type is intrinsically more applicable in the same manner as humans who can learn by simple visual observation. However, such a learning type is not well defined and, therefore, faces more challenges than the former frequently.

Normally, Machine Learning encompasses two stages. The first stage is the feature extraction stage. This stage involves extracting discriminative and informative subsets from supplied raw data. In the context of this work, Internet traffic is monitored and sent and received packets are collected. Also, these packets are examined and the necessary features are extracted to understand underlying traffic patterns. The second stage rallies around selecting a befitting model to process the extracted features to give the desired results. These models are often generated by the ability to minimise the prediction error.

k-NN

The k-Nearest Neighbors (k-NN) is a traditional non-parametric method used for classification and regression Altman (1992). It is one instance-based learning or lazy learning, in which the training stores the samples and induces all the computation costs in test time to find the k-Nearest Neighbors.

Given a training dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ and a test sample x_0, the target is to predict where x_0 belongs. During the training process, the dataset $\mathcal{D}$ is loaded and stored. Then, at test time, we retrieve the k nearest neighbors from the training data where k is a pre-specified hyper-parameter In general, we may

apply the voting method to a classification task, where the label of x_0 is chosen as the most frequent class among the k nearest neighbors of x_0 as shown in Figure 2.13. Similarly, in the regression task, we may set the mean value of the labels of k nearest neighbors to be the label of x_0 Osterrieder (2023).

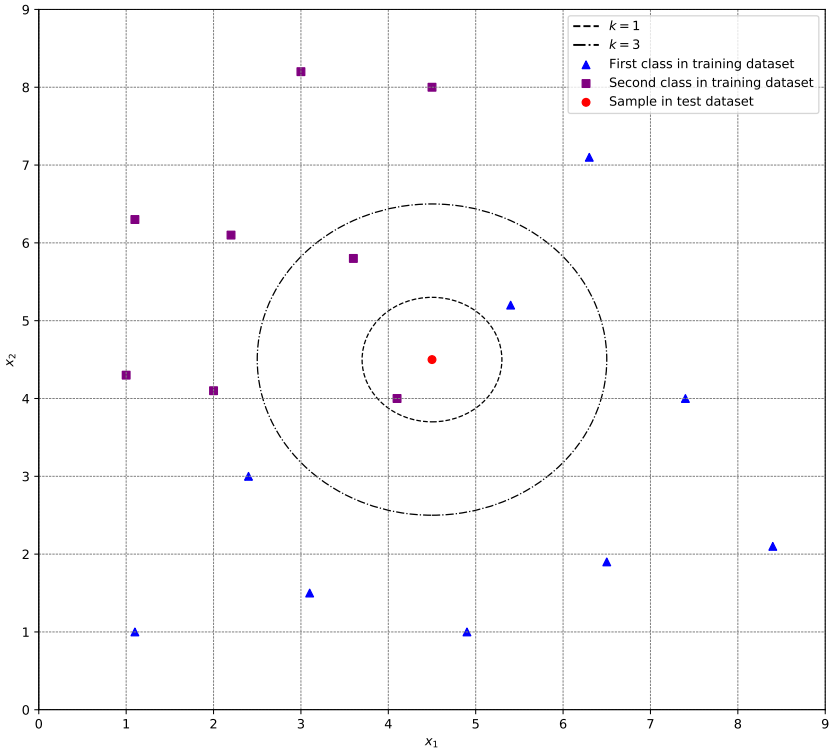

Figure 2.13: Illustration for k-NN when $k = 1$ and $k = 3$ Osterrieder (2023).

Obviously, the key factor of k-NN is to choose the k nearest neighbors. Generally, the distance metric is denoted as the *Euclidean distance*;

$$d(x_0, x_i) = \|x_0 - x_i\|_2 \tag{2.24}$$

where x_0 and x_i are points in the Euclidean space. These points can be vectors in

a multidimensional space. $\|x_0 - x_i\|_2$ denotes the Euclidean norm (or 2-norm) of the vector difference $x_0 - x_i$. In simpler terms, the Euclidean distance between x_0 and x_i is the straight-line distance between these two points in the space. The 2-norm (or Euclidean norm) is the most common way to measure this distance.

Therefore, the k nearest neighbors can be selected based on the k minimal Euclidean distances from the training dataset $\mathcal{D}$. We denote the set of the k nearest neighbors by $\mathcal{D}_k \in \mathcal{D}$, that is, the number of $\mathcal{D}_k$ is k. Finally, we utilize the voting method or mean value to execute classification tasks or regression tasks. Here are some other commonly used distance formulas for k nearest neighbors;

$$\text{Manhattan Distance} = d(x, y) = \sum_{i=1}^{m} |x_i - y_i| \tag{2.25}$$

The Manhattan Distance calculates the sum of the absolute differences between the corresponding elements of two vectors x and y. In this formula, x_i and y_i denote the i-th elements of vectors x and y, m is the total number of elements in the vectors, and $|x_i - y_i|$ is the absolute difference between the i-th elements.

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.26}$$

The Minkowski Distance is a generalized metric for measuring the distance between two points x and y in a normed vector space. In this equation, x_i and y_i are the i-th elements of the vectors x and y, n is the total number of elements in the vectors, and p is the order parameter which defines the type of distance (for instance, $p = 1$ for Manhattan distance, $p = 2$ for Euclidean distance).

$$\begin{aligned} \text{Hamming Distance} = D_H &= \sum_{i=1}^{k} |x_i - y_i| \\ x = y \quad &\Rightarrow \quad D = 0 \\ x \neq y \quad &\Rightarrow \quad D \neq 1 \end{aligned} \tag{2.27}$$

The Hamming Distance measures the number of positions at which the corresponding elements of two vectors x and y differ. Here, x_i and y_i represent the i-th elements of the vectors x and y respectively, k is the total number of elements in the vectors, and $|x_i - y_i|$ is the absolute difference between the i-th elements (1 if they are different, 0 if they are the same).

To improve the performance of k-NN, a large margin nearest neighbor or neighborhood components analysis can be used to enhance accuracy Weinberger & Saul (2009). The weighted nearest-neighbor classifier takes into account the weights of all the neighbors to promote performance Samworth (2012). In contrast, condensed nearest neighbor reduces the dataset to lessen the amount of calculation Hart (1968).

SVM

Fundamentally, a Support Vector Machine (SVM) is one of the best tools to solve classification problems and regression problems. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ where $y_i \in \{-1, +1\}, i = 1, 2, \ldots, n$. SVM will harness the most robust hyperplane for dividing the sample space of different classes, as shown in Figure 2.14. In the case where a dataset is linearly separable, one may choose two parallel hyperplanes, $\mathbf{w}^T\mathbf{x} + b = -1$ and $\mathbf{w}^T\mathbf{x} + b = 1$ to separate the dataset through the support vectors, and then the separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ separates the two classes. The goal is maximizing

the distance between the support vector and the hyperplane by optimizing the parameters $\mathbf{w}$ and b.

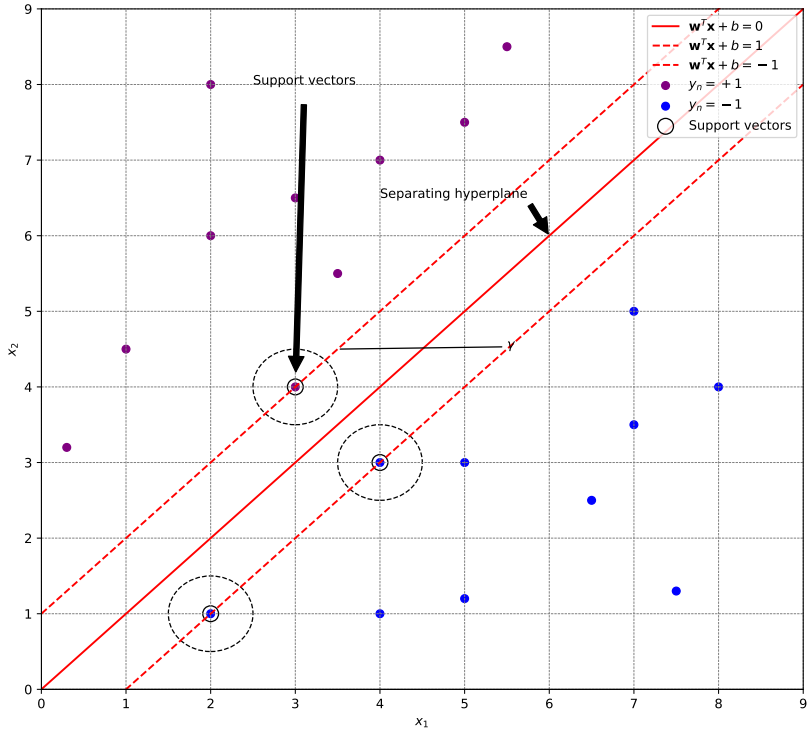

Figure 2.14: Illustration of hyperplane and support vector of SVM Osterrieder (2023).

The distance between a sample point $\mathbf{x}_n$ and hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ is denoted as $d_n = \frac{\mathbf{w}^T\mathbf{x}_n+b}{\|\mathbf{w}\|}$. Denoting by γ the distance between the support vector and hyperplane, optimization problem of SVM can be formulated as;

$$\begin{aligned} &\max_{\mathbf{w},b} \ \gamma \\ &\text{subject to} \ \ y_n\left(\frac{\mathbf{w}^T\mathbf{x}_n + b}{\|\mathbf{w}\|}\right) \geq \frac{\gamma}{2}, \ n = 1, 2, \ldots, N, \end{aligned} \tag{2.28}$$

where $\left(\frac{\mathbf{w}^T\mathbf{x}_n+b}{\|\mathbf{w}\|}\right)$ can be positive for $y_n = +1$ or negative for $y_n = -1$. Therefore, $y_n\left(\frac{\mathbf{w}^T\mathbf{x}_n+b}{\|\mathbf{w}\|}\right)$ must be positive and constrained to be greater than $\frac{\gamma}{2}$. To simplify the problem, we use the fact that $\gamma = \frac{2}{\|\mathbf{w}\|}$, which is the distance between $\mathbf{w}^T\mathbf{x} + b = 0$ and $\mathbf{w}^T\mathbf{x} + b = 1$. Thus problem 2.28 is equivalent to

$$\begin{aligned} &\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \ \ y_n(\mathbf{w}^T\mathbf{x}_n + b) - 1 \geq 0, \ \ n = 1, 2, \ldots, N. \end{aligned} \tag{2.29}$$

Problem 2.29 is the basic formula of SVM, which can be solved by using the convex optimization software CVX Grant & Boyd (2014). To solve the problem more efficiently, the dual problem of 2.29 can be exploited. The Lagrangian function can be written as

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{n=1}^{N} \alpha_n \left(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b)\right). \tag{2.30}$$

By equating the partial derivatives $L(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b to 0 we have

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n, \quad 0 = \sum_{n=1}^{N} \alpha_n y_n. \tag{2.31}$$

Therefore the dual problem of 2.29 can be expressed as

$$\begin{aligned} &\max_{\alpha} \ \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n\alpha_m y_n y_m \mathbf{x}_n^T\mathbf{x}_m \\ &\text{subject to} \ \ \sum_{n=1}^{N} \alpha_n y_n = 0, \\ &\alpha_n \geq 0, \ \ n = 1, 2, \ldots, N, \end{aligned} \tag{2.32}$$

which is a quadratic programming problem and can be efficiently solved by several optimization algorithms like sequential minimal optimization Platt (1998).

Finally, the linear model of SVM is given by

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n^T \mathbf{x} + b. \tag{2.33}$$

In the case of a dataset that is not linearly separable, sample features can be projected into some higher space than the original space, i.e., $\mathbf{x}' = \phi(\mathbf{x})$. This higher-dimensional $\mathbf{x}'$ may then be linearly separable, which could be used to determine the separating hyperplane. Similarly, the dual problem for the higher dimensional dataset $\mathbf{x}' = \phi(\mathbf{x})$ can also be expressed as:

$$\begin{aligned} &\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) \\ &\text{subject to } \sum_{n=1}^{N} \alpha_n y_n = 0, \\ &\alpha_n \geq 0,\ n = 1, 2, \ldots, N, \end{aligned} \tag{2.34}$$

and the linear model of the higher-dimensional dataset is

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{n=1}^{N} \alpha_n y_n \phi(\mathbf{x}_n)^T \phi(\mathbf{x}) + b. \tag{2.35}$$

Obviously, the key challenge is to calculate $\phi(\mathbf{x})^T\phi(\mathbf{x}_n)$, which may be extremely high-dimensional. To avoid this challenging problem, the kernel function is proposed as

$$k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m). \tag{2.36}$$

which can be well designed to reduce the computational complexity. The commonly adopted kernel functions are listed in Table 2.1.

More than that, the soft-margin method allows mapping a data set into a

Kernel Function	Formulation
Linear	$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^T \mathbf{x}_m$
Polynomial (homogeneous)	$k(\mathbf{x}_n, \mathbf{x}_m) = (\mathbf{x}_n^T \mathbf{x}_m)^d, d \geq 1$
Polynomial (inhomogeneous)	$k(\mathbf{x}_n, \mathbf{x}_m) = (\mathbf{x}_n^T \mathbf{x}_m + c)^d, d \geq 1$
Gaussian	$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_m\|^2}{2\sigma^2}\right), \sigma > 0$
Laplace	$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_m\|}{\sigma}\right), \sigma > 0$
Sigmoid	$k(\mathbf{x}_n, \mathbf{x}_m) = \tanh(\beta \mathbf{x}_n^T \mathbf{x}_m + \theta), \beta > 0, \theta < 0$

Table 2.1: Commonly adopted kernel functions.

higher dimension, though not high enough, while allowing the linearly separable property by tolerating errors on some samples. The hinge loss function is adopted to measure the degree of dissatisfaction of the constraint 2.29, which is a hard margin. Hence, the optimization problem 2.29 may be relaxed as follows;

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{n=1}^{N} \max(0, 1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)), \tag{2.37}$$

which is convex and can be solved by using the gradient descent or other optimization algorithm Boyd & Vandenberghe (2004).

Random Forest

Random Forest (RF) is an ensemble learning algorithm for classification and regression tasks. The model functions by building many Decision Trees in such a way that it will output either a class with the majority in classification or mean prediction from individual trees in the case of regression. A Decision Tree consists of three parts: internal nodes, leaf nodes, and branches. The

interior nodes represent the comparison of a single input within a record to a given value, and the leaf nodes represent the labels. The branches represent the conjunctions of features and the structure topology of the tree. In addition, a root node represents the particular, unique interior node at the tree's Decision Tree's entry. The typical structure of a Decision Tree is as shown below in Figures 2.15.

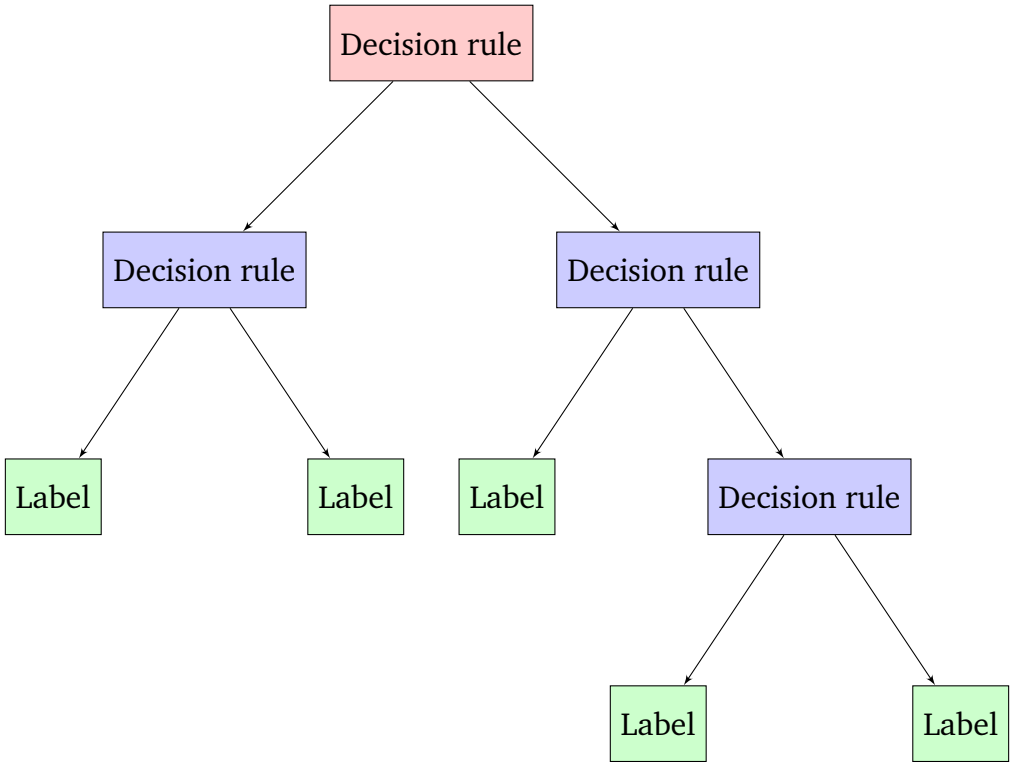

Figure 2.15: Decision tree structure

Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ where x_i represents the feature vector and y_i the target variable, the Random Forest algorithm works as follows:

1. **Bootstrap Sampling:** Generate B bootstrap samples from the original dataset D. Each sample is created by randomly selecting instances from D with replacement.

2. **Tree Construction:** For each bootstrap sample, train an unpruned Decision Tree using the Classification and Regression Tree (CART) algorithm Breiman (2017). In the case of classification, the splitting criterion of the CART is to minimize the Gini coefficient of the dataset. More precisely, consider a training set $\mathcal{D}$ having N samples, and K categories we define the kth category has N_k samples, that is, $N = \sum_{k=1}^{K} N_k$. Therefore the Gini coefficient of $\mathcal{D}$ is defined as;

$$\text{Gini}(\mathcal{D}) = \sum_{k=1}^{K} \frac{N_k}{N}\left(1 - \frac{N_k}{N}\right) = 1 - \sum_{k=1}^{K}\left(\frac{N_k}{N}\right)^2 \tag{2.38}$$

which reflects the purity of dataset, that is, the probability of having different labels for two randomly chosen samples.

In the root node, CART will benefit from a partition rule to split the training dataset $\mathcal{D}$ into several partitions. We specify the feature as x_j with possible feature values, that is, $x_j \in \{x_{j1}, x_{j2}, \ldots\}$. Hence, the dataset where the feature x_j of all samples is x_{ji} is denoted by $\mathcal{D}_{x_j=x_{ji}}$ and the number of samples is $N_{x_j=x_{ji}}$. The Gini coefficient of feature x_j in $\mathcal{D}$ is presented as

$$\text{Gini}_{\text{feature}}(\mathcal{D}, x_j) = \sum_{i} \frac{N_{x_j=x_{ji}}}{N} \text{Gini}(\mathcal{D}_{x_j=x_{ji}}) \tag{2.39}$$

Therefore the optimal partition feature in the root node is selected from

$$x_j = \arg\min_{x_j} \text{Gini}_{\text{feature}}(\mathcal{D}, x_j) \tag{2.40}$$

By repeating the mentioned process to decide other internal nodes the leaf nodes are generated when the Gini coefficient of a subdataset is 0. Finally, the Decision Tree is constructed.

3. **Aggregation:** For classification, aggregate the predictions of all trees and assign the class that receives the majority vote. For regression, compute the average of the predictions from all trees.

The prediction for a new instance x is given by:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{2.41}$$

where $T_b(x)$ is the prediction of the b-th tree.

C4.5

C4.5 is the decision tree generation algorithm developed by Ross Quinlan and an extension of his earlier ID3 Quinlan (1986) algorithm Quinlan (2014). C4.5's generated Decision Trees can perform a classification, so this is often called a statistical classifier. Given a training dataset, S with N samples, each having M attributes, such that S is composed of samples from K distinct classes, then the C4.5 algorithm functions as follows:

- **Splitting Criteria:** The C4.5 Decision Tree selects the attribute at each node that best divides the set of samples into subsets that are more abundant in one class or another. The criterion for splitting is based on the concept of information gain ratio, which can be calculated using the entropy measure from information theory. The information gained for an attribute A is defined as:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \tag{2.42}$$

where Entropy(S) is the entropy of the original dataset S, and S_v is the subset of S for which the attribute A has value v. The entropy is calculated as:

$$\text{Entropy}(S) = -\sum_{i=1}^{K} p_i \log_2(p_i) \tag{2.43}$$

where p_i is the proportion of samples in class i. The gain ratio is then calculated to normalize the information gain:

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)} \tag{2.44}$$

where

$$\text{SplitInformation}(S, A) = -\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right) \tag{2.45}$$

- **Handling Continuous Attributes:** C4.5 can handle both continuous and discrete attributes. Regarding constant attributes, the algorithm defines a threshold value at which the gain on information is maximized; afterward, data gets divided into two subsets using the attribute values above the threshold and values equal to or below it.

- **Pruning:** After the original tree has been built, C4.5 prunes low importance branches. This simplification step can reduce the overfitting of the final decision tree so that it will generalize better to unseen data. Pruning refers to replacing a subtree with a leaf node when the subtree does not help in making correct classifications.

- **Handling missing values:** The C4.5 algorithm can be worked on with a probabilistic approach for missing attribute values. It will assign a probability to each possible attribute value, which are used in the calculations of information gain.

- **Rule Generation:** C4.5 can convert the generated Decision Trees into a set of if-then rules. In this way, better human understanding is facilitated since these are easier to understand than Decision Trees. Sometimes, rules prove more effective in a given data set.

The prediction for an instance x is based on the decision path taken within the tree, according to the values of its attributes up to the leaf node. That is, the class that pertains to such leaf nodes is what is predicted for x.

2.4.2 Deep Learning

Deep Learning (DL) is a form of Machine Learning that involves a model learning directly from raw data input without necessarily being subjected to the feature engineering phase. The term is often used essentially interchangeably with Artificial Neural Networks (ANN)s, as the earliest implementations bore only a loose resemblance to how the brain appeared to work biochemically Goodfellow et al. (2016). One of the most elementary types of ANN is the Multilayer Perceptron (MLP) Bishop (2006), which calculates a vector of outputs $y = [y_1 \dots y_K]^T$ through M linear combinations of elements of the input $x = [x_1 \dots x_D]$ and subsequent passage of the result through a non-linear activation function h:

$$a_j = \sum_{i=1}^{D} W_{j,i}^{(1)} x_i + b_j^{(1)}, \quad j = 1, \dots, M \tag{2.46}$$

$$z_j = h(a_j), \tag{2.47}$$

where $W_{j,i}$, and b_j are trainable network parameters, i.e., weights and biases. The superscript refers to the first layer of the MLP. For the simplest case where

the ANN has only one hidden layer: the output activations are just directly obtained from z_j through another linear combination:

$$a_k = \sum_{j=1}^{M} W_{kj}^{(2)} z_j + b_k^{(2)}, \quad k = 1, \ldots, K \tag{2.48}$$

Depending on the type of responses, output activations can be fed through another one of the many nonlinear activation units which will be explained later or taken as identity $y_k = a_k$ (see Figure 2.16). Due to the development of efficient optimization techniques and advances in the computational capability of consumer-grade computers, mainly regarding Graphics Processing Units (GPUs), models with a more significant number of hidden layers, i.e., more depth, were capable of being trained within reasonable timespans, prompting a resurgence of research in the late 2000s, hence popularizing the use of deep learning, or Deep Neural Network (DNN)s, as synonyms for neural networks.

Similarly to classical ML, the training process of a DNN is formulated in terms of minimization of a scalar loss function $f(x, y, \theta)$; here, θ denotes the set of learnable parameters of the network, and the dependence on the inputs and outputs is made explicit. As a DNN is nonlinear by design (recall Eq. 2.47), finding the optimal θ is achieved through iterative methods by searching for critical points in f using the gradient information; the most straightforward approach is taking a small step in θ-space in the direction of the negative gradient, i.e., gradient descent Bishop (2006):

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_k \nabla_\theta f, \tag{2.49}$$

where α_k is the learning rate at time-step $t = t_k$.

One pass through the DNN is sufficient for computing the conditions, which,

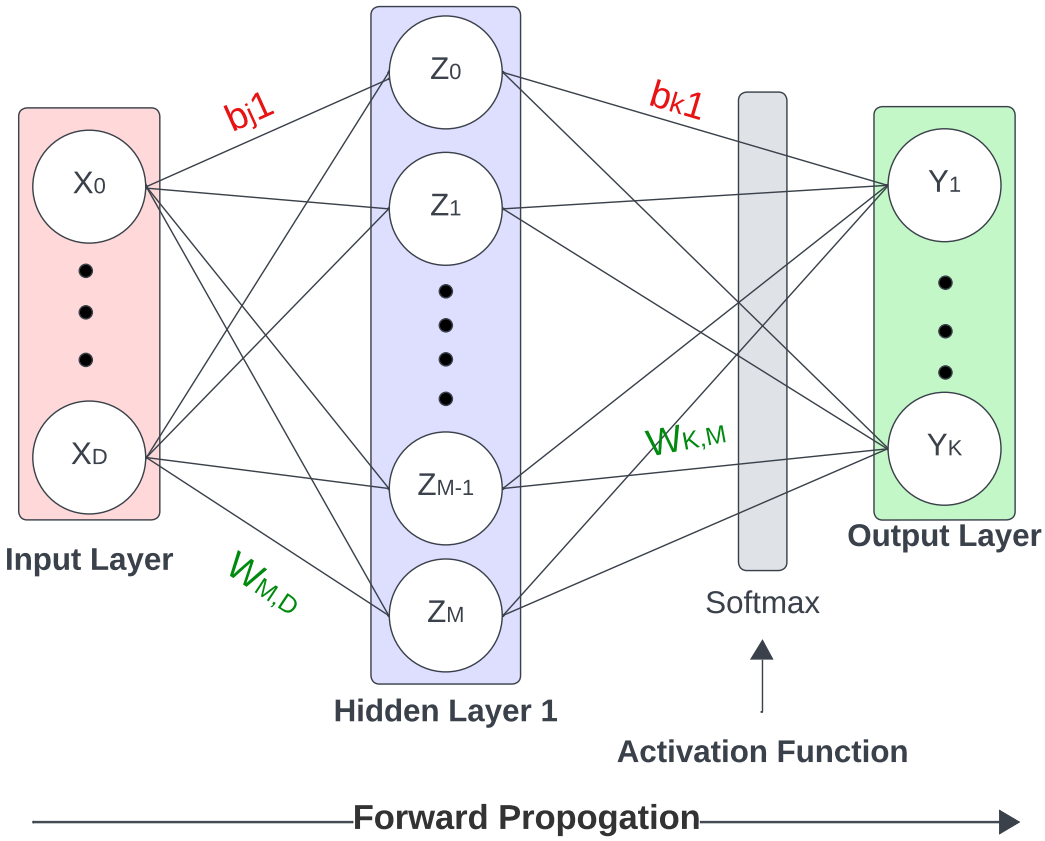

Figure 2.16: Diagram for a two-layer Artificial Neural Networks (ANN). The open nodes represent input (x), hidden (z), and output (y) variables, whereas weights (W) are represented by links connecting nodes. Biases (b) are also represented as links which come from additional closed nodes ($x0$, $z0$). Cf. Equations 2.46 to 2.48. Adapted from Bishop (2006).

together, make it possible to calculate the gradient of the loss concerning the output. That is, $\nabla_y f$. The gradient of f concerning the weights in each layer, required for the gradient descent optimization in Equation 2.49, is obtained by successively chaining the local gradients of each layer in the reverse direction until the desired layer is reached (backpropagation; Rumelhart et al. (1986)). If a unit j in one layer sends connections to k units in the next layer, then the local gradient at j is:

$$\frac{\partial f}{\partial a_j} = \sum_k \frac{\partial f}{\partial a_k} \frac{\partial a_k}{\partial a_j}. \tag{2.50}$$

Therefore, it becomes possible to calculate the gradients for all hidden layers with recursive backpropagation efficiently, regardless of the DNN model, since the forward pass gives the gradient values for the output units.

Since gradient-based parameter optimization very often becomes prohibitive given memory requirements for huge datasets, we can sample it at each step $\mathbb{B} = \{x^{(1)}, \ldots, x^{(m)}\}$ a mini-batch of m inputs from the training data set, and the task will be done in the same way. In this case, the procedure becomes a Stochastic Gradient Descent (SGD), and the gradient is estimated as:

$$\nabla_\theta f \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta f \left(x^{(i)}, y^{(i)}, \theta \right). \tag{2.51}$$

An epoch passes once the SGD processes all minibatches comprising the entire dataset of inputs. Typically, a DNN requires several epochs to train.

A linear output, as given in Equation 2.48, permits a neural network to learn regression problems. However, much of the popularity exuded by Deep Learning today comes from its potential performance for classification tasks—i.e., a discrete and generally mutually exclusive output—in particular those where a

selection is performed from a large pool of choices. Where the sigmoid function (see Figure 2.17, middle) is used to model output that follows a Bernoulli distribution (like in a coin toss, or in telling a dog from a cat, or distinguishing between low-latency requiring traffic and others), a generalization of the sigmoid function is what gets called the softmax function, defined as:

$$\text{softmax}(\mathbf{a})_k := \frac{\exp(a_k)}{\sum_\ell \exp(a_\ell)}, \tag{2.52}$$

is used to model categorical, or generalised Bernoulli, distributions (e.g. distinguishing between several breeds of cat, or differentiating between multiple Internet traffic types).

Activation Functions

The earliest ANNs generally used either sigmoid or hyperbolic tangents for the activation function (middle and left, resp., of Figure 2.17) Goodfellow et al. (2016). In both cases, the function saturates for most of its domain, being sensitive to an input only near zero, which makes learning through vanishing gradients a complex phenomenon during backpropagation. Despite that, they were initially used in a widespread manner and are nowadays discouraged. The latter are usually easier to train, but they also depend on small activation inputs.

A popular modern default choice for the activation function of a node is the so-called rectified linear unit (ReLU), as defined next:

$$relu(x) := max(0, x) \tag{2.53}$$

The function's response is to simply output zero if the input is negative and be linear when the input is positive (Figure 2.17, R); it is thus easy to optimize

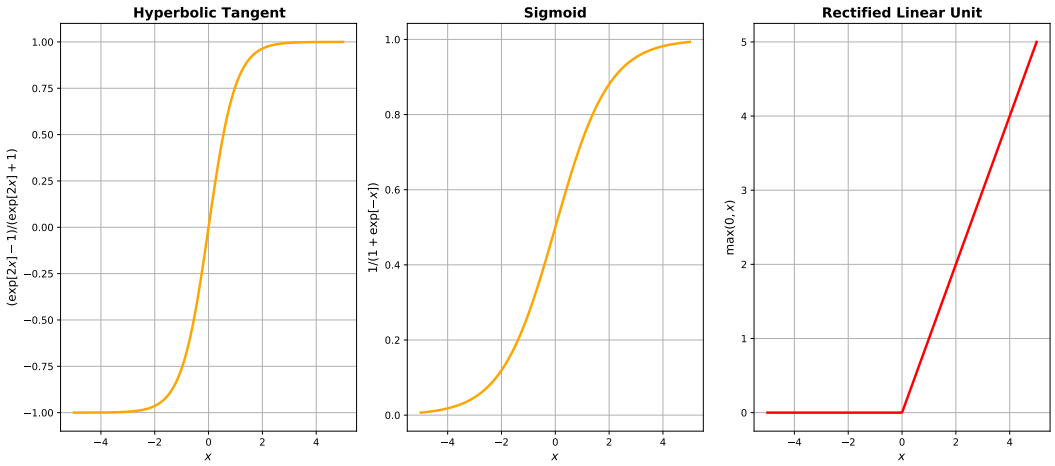

Figure 2.17: Common Artificial Neural Networks (ANN) non-linear activation functions. L(eft): $h(x) = tanh(x)$, M(iddle): $h(x) = sigm(x)$, R(ight): $h(x) = relu(x)$, respectively

for this reason. Other advantages include sparse activation, with about 50% of the hidden units having a non-zero output for a randomly initialized ANN—good to prevent overfitting and hence generalizing well to new examples—and better gradient propagation when compared to the sigmoid and hyperbolic tangent functions that saturate in both directions.

Although the ReLU activation has many advantages, it could run into specific problems during backpropagation, mainly resulting in convergence to states of inactivity regardless of the input. This problem has been widely referred to as the 'dying ReLU' issue and occurs when a huge negative bias is learned by the network. So, in driving the output of the activation (in this case, an activation's output) down to zero (see Figure 2.17, R), backward gradient flow is effectively blocked. To make this more stable, a couple of generalizations of the ReLU function have been proposed. One of these is very well known, which is the

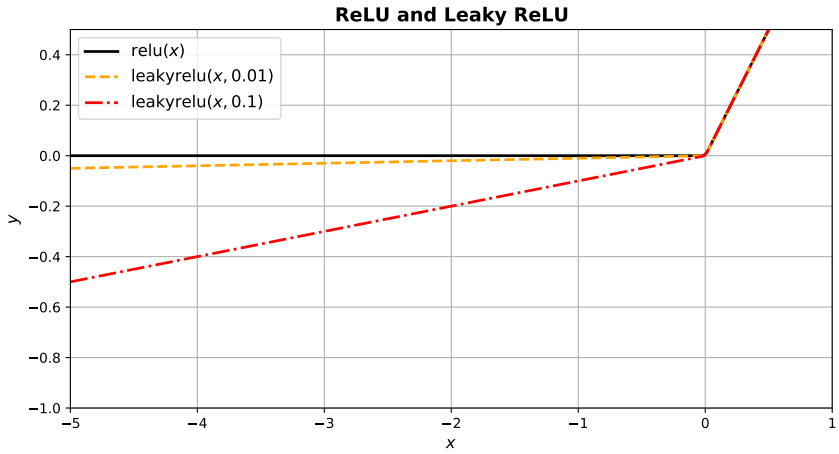

Figure 2.18: Comparison of the ReLU activation function with the leaky ReLU activation function. The latter is plotted for two different values of η, the parameter that determines the slope of the response when $x < 0$.

leaky ReLU Maas et al. (2013), where the output gets scaled up if the unit is not active:

$$\text{leakyrelu}(x, \eta) := \begin{cases} x & \text{if } x > 0, \\ \eta x & \text{otherwise.} \end{cases} \tag{2.54}$$

The function is illustrated in Figure 2.18. Also, different solutions exist, such as the parametric ReLU, which treats η as a learnable parameter He et al. (2015).

Optimisation

The beginning of Section SGD introduced as the 'default' learning method for DNNs. Despite its simplicity, however, it usually leads to a slow learning process Goodfellow et al. (2016). To speed up the optimization, momentum can be applied to improve SGD by adding an auxiliary variable that plays the role

of velocity, which describes the direction and rate of progression throughout parametric space. The parameter update becomes (cf. Eqs. 2.49 and 2.60)

$$\mathbf{v}^{(k+1)} = \gamma \mathbf{v}^{(k)} - \alpha \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} f\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \boldsymbol{\theta}\right) \right) \tag{2.55}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{v}^{(k+1)} \tag{2.56}$$

Equation 2.55 shows that SGD with momentum preserves the effect of past steps by maintaining a moving average of previous gradients, where $\gamma \in [0, 1]$ is the decay rate. Momentum is essential in situations where the Hessian matrix is poorly conditioned; for instance, where the objective function's topology is highly variable in most directions perpendicular to the path towards a minimum, regular SGD would be adversely affected.

However, momentum does not solve the problem of selecting the probably most relevant hyperparameter (the learning rate) but introduce one more. The solution is to define optimization methods in which the learning rates of model parameters get adapted. The algorithm RMSProp, unbublished and introduced by Geoffrey Hinton in 2012 during his lectures Hinton (2012), simply scales each parameter learning rate in cector θ inversely in proportion to the past accumulated squared gradient. As the accumulation comes through an exponentially decaying average, the influence of the distant past is made neglibible while ensuring rapid congergence when convex bowl is obtained Goodfellow et al. (2016). The update is given by:

$$\mathbf{g}_{\text{acc}}^{(k+1)} = \gamma \mathbf{g}_{\text{acc}}^{(k)} + (1 - \gamma)\mathbf{g}^{(k+1)} \odot \mathbf{g}^{(k+1)}, \tag{2.57}$$

$$\Delta\boldsymbol{\theta}^{(k+1)} = -\frac{\alpha}{\sqrt{\mathbf{g}_{\text{acc}}^{(k+1)} + \epsilon}} \odot \mathbf{g}^{(k+1)}, \tag{2.58}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}^{(k+1)}, \tag{2.59}$$

Where $\odot$ is an element-wise product, ϵ is a small constant to stabilize divisions by potentially small numbers, the divisions, and the square root are applied element-wise, and g is the gradient.

The Adam algorithm by Kingma & Ba (2014) improves upon this by adding another term, similar to momentum (cf. Eq. 2.55) but based upon the non-squared gradient, and then adds a bias correction estimate to both of the terms dependent on the gradient. The overall update may be summarised as:

$$\mathbf{g}_{\text{1st}}^{(k+1)} = \gamma_1 \mathbf{g}_{\text{1st}}^{(k)} + (1-\gamma_1)\mathbf{g}^{(k+1)}, \tag{2.88}$$

$$\mathbf{g}_{\text{2nd}}^{(k+1)} = \gamma_2 \mathbf{g}_{\text{2nd}}^{(k)} + (1-\gamma_2)\mathbf{g}^{(k+1)} \odot \mathbf{g}^{(k+1)}, \tag{2.89}$$

$$\hat{\mathbf{g}}_{\text{1st}}^{(k+1)} = \frac{\mathbf{g}_{\text{1st}}^{(k+1)}}{1-\gamma_1^{k+1}}, \tag{2.90}$$

$$\hat{\mathbf{g}}_{\text{2nd}}^{(k+1)} = \frac{\mathbf{g}_{\text{2nd}}^{(k+1)}}{1-\gamma_2^{k+1}}, \tag{2.91}$$

$$\Delta\boldsymbol{\theta}^{(k+1)} = -\frac{\alpha}{\sqrt{\hat{\mathbf{g}}_{\text{2nd}}^{(k+1)} + \epsilon}} \odot \hat{\mathbf{g}}_{\text{1st}}^{(k+1)}, \tag{2.92}$$

Where the divisions and square root are used element-wise, and the parameter update is the same as in Equation 2.59. It can also be noted in the bias

update operations that the decay parameters γ_1 and γ_2 are exponentiated by the current iteration value k. In this respect, Adam provides more robustness to hyperparameter choice than SGD with momentum RMSProp.

Transfer Learning

DNNs that translate into large models might need not only long training times but also massive training datasets. A possible approach toward mitigating these two obstacles is transfer learning: the assumption that some factors responsible for influencing the outcome of one task are relevant to the result of a different task. Concretely, in the case of CNNs, it is expected that several of the learned kernels converge toward detecting generalized visual features. This has been experimentally validated on image classification for tasks where the first layers' convolution kernels are optimized toward wide-domain and broad features, like corners and edges. In contrast, the last layers' kernels specialize in more problem-specific shapes or other structures related to the problem at hand Zeiler & Fergus (2014). Thus, it is expected to adopt a pre-existing CNN architecture as the backbone of the model and retrain on general datasets. The early layers can be trained on a huge general dataset, up to some layers. Then, fewer later layers are trained from scratch using more minor domain-specific data.

One of the most well-known datasets for visual tasks in transfer learning is the ImageNet dataset, which comprises over 14 million hand-annotated images in bins with over 20,000 categories through crowd-sourcing. Many of the critical advancements in the last decade in designing the architecture of CNNs come from participation in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a challenge whose objective is to build an ML pipeline to classify

images into 1000 different classes of ImageNet correctly. Significantly, DNNs' trampoline jump into the public view began with Krizhevsky's Krizhevsky et al. (2012) participation in the ILSVRC 2012 contest, in which a variant of LeCun's Lecun et al. (1992) own CNN—this time trained on a GPU that massively accelerated the process—won by achieving a top-5 classification error of 15.3%, more than almost 11 p.p. lower than the runner-up. Since then, new CNN designs have competed in the challenge each year, resulting in successive dramatic improvements while showcasing new advances, such as GoogLeNet Szegedy et al. (2015), which was characterized not only by a profound architecture but also by parallel layers to detect features at different scales; and ResNet He et al. (2015), which introduced residual connections allowing the breakthrough to even deeper architectures. Many of these state-of-the-art DNN architectures have been open-sourced, and their ImageNet pre-trained weights have been made available, contributing to rapid progress in the field and the adoption of such models as a CNN front-end.

Regularisation

The number of learnable parameters relates to the capacity of a DNN model and becomes an implicit hyperparameter. A larger capacity will yield a better potential for DNNs to predict with lower errors. If the number of parameters is too high, it can lead to overfitting, meaning that the network, in fact, stops learning but starts to memorize training data and hence cannot generalize new unseen test data. For this reason, it is expected to assess the network's performance on a validation set during the training process. A validation set is not employed in the actual training process itself but only for the sake of insight into the gen-

eralization ability of the pipeline. In particular, when the validation error is much higher than the training error, this could indicate overfitting. Conversely, too few parameters may lead to underfitting, characterized by high training and validation errors.

To avoid overfitting, which is achieved by the reduction in the number of parameters, we could implement early stopping, consisting of stopping the learning process when the validation error starts to increase away from the training error. Furthermore, regularization procedures, such as adding noise to the learning process, can be introduced. In other words, some kind of regularization should always be used unless the training set is enormous—on the order of tens of millions of examples. This subsection describes several popular regularization techniques for DNNs Goodfellow et al. (2016).

Dropout Dropout Hinton et al. (2012) is, in effect, a procedure that randomly removes units from a network, namely by multiplying the output value of these units by zero. Dropout is an attempt to approximate the ML concept of bagging—that is, training K different models for K different training data subsets—in the case of intense networks. In each step of the minibatch training, one would sample a binary mask for every layer's hidden units, with application probability p. This probability is fixed and taken as a hyperparameter of each layer. Typical values are $p = 0.5$ for FC layers and $p\epsilon \in [0.1, 0.2]$ for convolutional layers placed before the activation function.

Weight Decay Weight decay is just about the oldest form of regularization there is in machine learning, and that goes back farther than the ANNs. It is based on adding a term to the loss function proportional to the weights of each

layer. A common weight decay type is L^2 regularization, which adds the squared sum of weights. Biases are not usually considered for weight decay. This leads to the following modification of the gradient of the $i-th$ layer Goodfellow et al. (2014):

$$\nabla_W f_i\left(W^{(i)}, x, y\right) \leftarrow \lambda W^{(i)} + \nabla_W f_i\left(W^{(i)}, x, y\right), \tag{2.60}$$

Where λ is a hyperparameter typically chosen on a logarithmic scale, say $\lambda\epsilon\{10^{-6}, 10^{-5}, ..., 10^{-2}\}$. Despite this longstanding history of weight decay in the context of ML, for modern CNNs, alternative strategies are typically employed additionally or as a complete alternative.

Batch Normalisation To normalize inputs for a layer, Ioffe & Szegedy (2015) computed the mean μB and variance σ_B^2 of each mini-batch m:

$$a^{(i)'} \leftarrow \frac{a^{(i)} - \mu_{B_m}}{\sqrt{\sigma_{B_m}^2 + \epsilon}}, \quad a^{(i)'} \Leftarrow x^{(i)} \in B_m = \{x^{(1)}, \ldots, x^{(M)}\}, \tag{2.61}$$

where ϵ is a small numerically stabilising term. Furthermore, the outputs are modified by a learnable scale and offset:

$$a''^{(i)} \leftarrow \gamma_{B_m} a'^{(i)} + \beta_{B_m}. \tag{2.62}$$

The average mean and variance of the entire data set are approximated at training time via the moving average of each per-batch $\{\mu B_m, \sigma_{B_m}^2\}$, which is then used in input normalization. The batch norm was designed to improve the optimization of DNN in general, but it introduces noise into the system, possibly regularizing it.

CHAPTER 3

Related Work

This chapter provides an overview of recent developments in Internet traffic classification, focusing on the use of advanced machine learning techniques for identifying low-latency traffic. It explores key methods such as ANN, Wavelet Transforms, and trend-based feature extraction. The chapter highlights the both advantages and disadvantage of traditional approaches like deep packet inspection, sniffing and port-based methods, emphasizing the advantages of modern, more adaptive techniques. It also reviews alternative approaches and their performance in addressing the growing complexity of network traffic classification in both encrypted and real-time environments.

3.1 Traditional Methods for Internet Traffic Classification

3.1.1 Deep Packet Inspection

Deep Packet Inspection is an advanced form of network traffic analysis that looks at the inside contents of packets of data beyond just the header information. Unlike traditional packet analysis methods, which tend to focus on packet headers such as the IP and TCP layers, DPI delves into the application data layer, encompassing the top three layers (Session, Presentation, and Application) of the OSI model. Such deep inspection enables the network administrator to identify, classify, and manage network traffic more precisely, detecting specific applications and advanced threats within the data payload Finsterbusch et al. (2013). Figure 3.1 shows how the process of DPI works compared to the traditional packet analysis.

Packet capture, via network taps, mirror ports, or dedicated capture devices, is where the process of a DPI begins. While the packet capture in smaller networks can be performed with the help of tools like Wireshark, larger implementations would call for more advanced solutions that could cope with much higher data volumes and speed Khazankin et al. (2017). After packet capture, identification of protocols by header inspection of packets in the DPI system is necessary to identify the transport layer protocol utilized by the packet, which may range from TCP to UDP and others, and payload analysis for defining application-layer protocols including HTTP, HTTPS, FTP, and DNS.

In DPI, all the payload is examined using pattern matching and/or signature

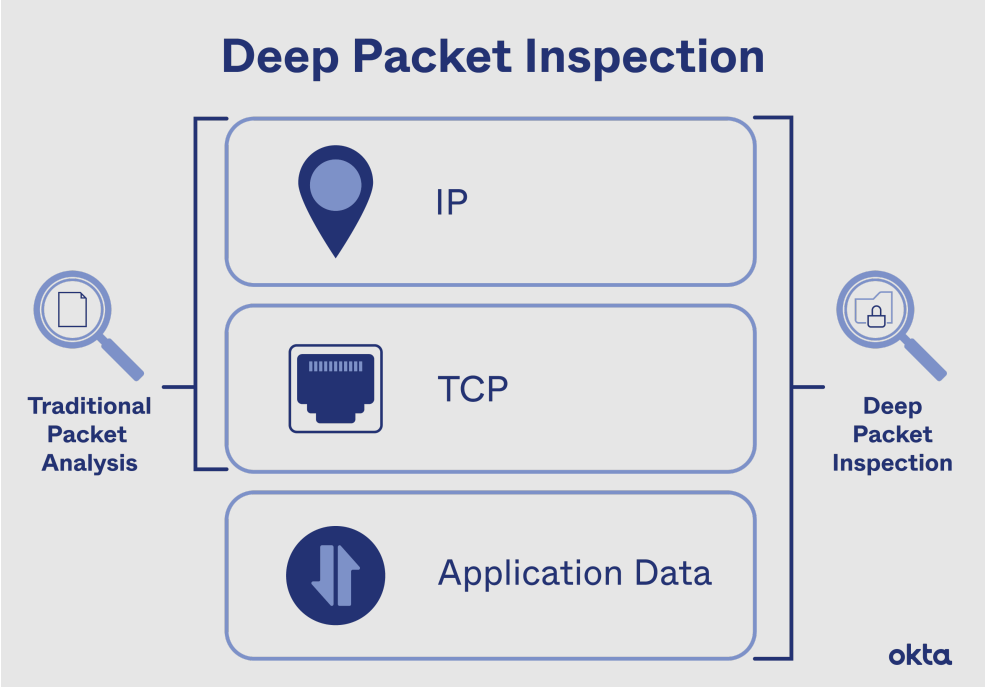

Figure 3.1: Process of DPI compared to traditional packet analysis Okta (2024).

analysis. This method is used by matching the contents of a payload against a database of known application, malware, and other types of traffic signs Yan et al. (2019). The methods typically adopted are regular expressions and heuristic analysis. The pattern matching process can be mathematically represented as :

$$\text{Match}(P, S) = \sum_{i=1}^{n} \delta(p_i, s_i) \tag{3.1}$$

where P is the pattern, S is the payload string, p_i and s_i are characters in the pattern and payload respectively, and δ is an indicator function that returns 1 if the characters match and 0 otherwise.

Besides pattern matching, the DPI systems carry out a behavioral anomaly analysis in searching out possible security threats. This takes the form of analyz-

ing flows of traffic to detect unusual port usage or unknown traffic volume or, for that matter, deviation from typical usage patterns Palmieri & Fiore (2010). Example being at the packet arrival rate, which is analyzed using statistical means of mean and variance:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{3.2}$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2 \tag{3.3}$$

where X_i represents the inter-arrival times of packets.

In addition, the DPI examines in detail the real content of the payload, which could be in text form, images, videos, or even executable files. This provides for content filtering, sensitive information detection, detection of embedded malware Lin et al. (2014). The inspection over the payload can be modeled as a multi-stage classification problem, where more detailed inspections refine the classification.

While DPI provides pervasive benefits in network traffic management and security, it also involves a number of challenges and limitations.

- **Processing Overhead and Performance:** The process of inspecting the payload of each packet introduces heavy computational overhead, which might lead to latency and degraded performance, particularly in high-speed networks Melo et al. (2014). Regarding this aspect, several kinds of research have been carried out for accelerating processing with hardware support via Field-Programmable Gate Array (FPGA)s,GPUs, and parallel processing techniques Ceška et al. (2019), Lin et al. (2016).

- **Encrypted Traffic:** The increasing deployment of encryption protocols such as TLS/SSL and HTTPS renders traditional DPI ineffective, since the payload is not accessible anymore for the inspection process Papadogiannaki & Ioannidis (2021). The TLS/SSL interception solutions, which are based on man-in-the-middle approaches, enable the decryption of the traffic; however, at a serious cost regarding privacy and legal issues Waked et al. (2020). Other approaches, such as statistical traffic analysis and machine learning, are currently investigated, trying to classify encrypted traffic without the need for decryption Niu et al. (2019).

- **Evasion Techniques:** Attackers use various evasion techniques like obfuscation, tunneling, and protocol impersonation to avoid detection by DPI Ishikura et al. (2021). This involves constant updates to the DPI system with the integration of sophisticated algorithms for detection.

Public concern over DPI raises pertinent ethical and legal questions regarding privacy and data protection.

- **Public Privacy Concerns:** It has been disclosed regarding packet payload inspection technology that sensitive personal information might be collected in ways that are likely to infringe on privacy rights Daly (2011). This can undermine users' trust in network services; in this case, users may not even be aware that their data is being inspected.

- **Regulatory Compliance:** Expressive laws further enhance demands towards data processing and privacy—such as, for instance, the General Data Protection Regulation of the European Union (GDPR) Geko & Tjoa

(2018). Organizations have to ensure their use of DPI is compliant with such regulations; this can be through user consent or data anonymization.

- **Ethical Debates:** The balance of security with privacy is a key ethical concern. While DPI can make the network more secure by detecting the threats, it may be employed for surveillance and censorship, converting into an infringement of freedom of individuals Corwin (2011).

Following are some areas where DPI is applied.

- **Network Security:** It improves intrusion detection systems and intrusion prevention systems through the detection of complex attack patterns and zero-day exploits that cannot be detected using header analysis alone Sun et al. (2011).

- **Traffic Management and Quality of Service:** DPI enables ISPs and network administrators to implement bandwidth allocation policies, give priority to life-critical applications, and allows automatic enforcement of network policies according to a variety of parameters including user identity, application type, location, time of day, and endpoint device type Bahaa et al. (2020).

- **Content Filtering and Compliance:** DPI can also be utilized to attain Acceptable Use Policy (AUP) enforcement and forbid access to content that is not allowed Mueller & Asghari (2012). It ensures compliance of organizations with the legislative requirements. This could be one of the reasons why governments might, within their censorship activities, consider resorting to this for which serious human rights concerns arise.

3.1.2 Packet Sniffing

Packet sniffing is a technique for network traffic analysis; it is the practice of intercepting and logging data as it flows over a network. This technique unveils excellent details about network communications, allowing network administrators to check traffic, find anomalies, troubleshoot problems, and improve security Oluwabukola et al. (2013), Qadeer et al. (2010), Kim & Reddy (2008). Operating primarily at Layer 2 (Data Link Layer) and Layer 3 (Network Layer) of the OSI model, packet sniffing plays a crucial role in both wired and wireless networks.

A network packet consists of a header and a payload. The header carries control information, including source and destination addresses, protocol identifiers, and sequencing information. The payload carries the actual data to be transmitted. For example, the Ethernet packet header, Figure 3.2, includes fields such as the destination MAC address, source MAC address, and EtherType, which specifies the protocol encapsulated in the payload Tanenbaum & Wetherall (2011).

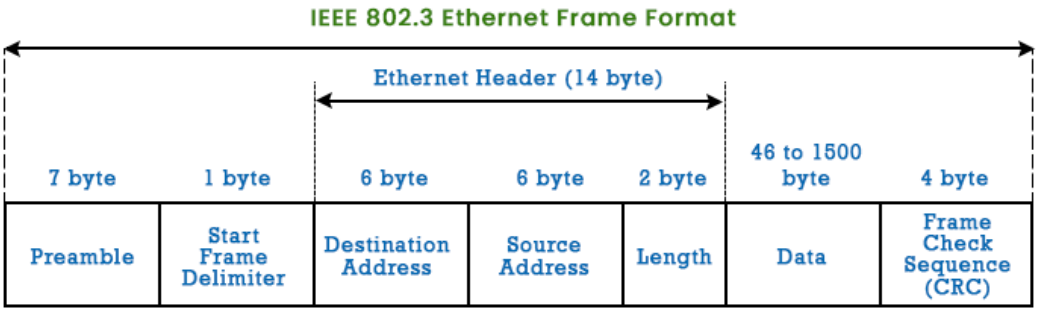

Figure 3.2: Structure of an ethernet packet.

Packet sniffing usually involves setting the Network Interface Card (NIC) into promiscuous mode. Hence, the NIC captures all the packets on the network segment, not just those addressed. This capability forms an essential foundation for a sniffer to monitor and analyze all traffic on a network Stallings (2015). However, operating in promiscuous mode can raise security concerns, as it allows potential eavesdropping on sensitive communications.

The first step in packet sniffing is capturing the packets. This is achieved using packet capture tools or libraries like libpcap on Unix-based systems and WinPcap on Windows. These libraries provide Application Programming Interface (API)s for low-level network access, enabling the capture of packets directly from the network interface Beale et al. (2006). While these libraries are widely used, they have limitations in high-speed networks due to potential packet loss at the kernel level Morariu & Stiller (2008). Alternatives like Data Plane Development Kit (DPDK) offer kernel bypass techniques to enhance performance but introduce increased complexity Ullah et al. (2020).

```
#include <pcap.h>


char errbuf[PCAP_ERRBUF_SIZE];
pcap_t *handle = pcap_open_live("eth0", BUFSIZ, 1, 1000,
    errbuf);
if (handle == NULL) {
    fprintf(stderr, "Could not open device: %s\n",
       errbuf);
    return(2);
}
```

Captured packets can be filtered to focus on specific traffic of interest. The Berkeley Packet Filter (BPF) syntax is commonly used for this purpose. Filters can be set to capture packets based on attributes such as IP addresses, port numbers, and protocol types. For example, a filter to capture TCP traffic on port 80 is written as:

```
tcp port 80
```

This filter is compiled and applied to the capture handle:

```
struct bpf_program fp;
if (pcap_compile(handle, &fp, "tcp port 80", 0,
   PCAP_NETMASK_UNKNOWN) == -1) {
    fprintf(stderr, "Could not parse filter: %s\n",
       pcap_geterr(handle));
    return(2);
}
if (pcap_setfilter(handle, &fp) == -1) {
    fprintf(stderr, "Could not install filter: %s\n",
       pcap_geterr(handle));
    return(2);
}
```

While BPF provides efficient filtering mechanisms, its syntax can be limiting for complex filtering requirements. Advanced packet processing frameworks like PF_RING offer enhanced filtering capabilities with better performance Ye et al. (2021).

Once packets are captured and filtered, they are analyzed to extract useful

information. This involves parsing the packet headers and payloads to identify protocols, reconstruct sessions, and detect anomalies. For example, analyzing a TCP packet involves extracting fields such as the sequence number, acknowledgment number, flags, and payload data.

```
const struct sniff_tcp *tcp;
tcp = (struct sniff_tcp*)(packet + SIZE_ETHERNET +
   size_ip);
printf("Source Port: %d\n", ntohs(tcp->th_sport));
printf("Destination Port: %d\n", ntohs(tcp->th_dport));
```

The increasing use of encryption protocols like TLS 1.3 indeed poses significant challenges to packet analysis, as payloads and even some header information are encrypted. This limitation has been acknowledged in multiple studies. For example, the encryption of internet traffic through TLS has made traditional content-based packet analysis yield diminishing returns, forcing the development of new approaches such as Machine Learning for traffic classification without access to payloads De Lucia & Cotton (2019).

Packet sniffing has become an integral component of network security to detect intrusion attempts, malware propagation, and DoS attacks, among others Dsouza et al. (2022). Generally, an analyst may trace back certain activities through traffic patterns and their contents that suggest a breach in security. However, dependency for security on packet sniffing is limited. Encryption prevents content inspection, while high volumes of data overload the analytical tools and often allow the threats to pass unnoticed De Lucia & Cotton (2019). Using packet sniffing, administrators are also able to monitor metrics regarding

network performance such as latency, throughput, and even packet loss Robles et al. (2021). This information is useful in finding faults in networks and in optimizing settings. The problem with this is, in high-speed networks, it is a hard job to capture and process packets in real time with no loss without special hardware or optimized software solutions Fais et al. (2021). New network technologies, such as SDN and network virtualization, change the face of network architectures. Packet sniffing should migrate to virtualized environments and integrate with SDN controllers and virtual switches Le & Thinh (2022). Traditional ways of sniffing may not work here, and a new approach will be required.

Tool	Interface	Protocol Support	Performance	Advanced Features
Wireshark	Graphical	Extensive	Moderate	Yes
Tcpdump	Command-line	Extensive	High	Limited
Tshark	Command-line	Extensive	Moderate	Yes

Table 3.1: Comparison of packet sniffing tools.

Several tools are widely used for packet sniffing and comparison of them demonstrated in Table 3.1:

- **Wireshark:** An open-source network protocol analyzer that provides a graphical interface for capturing and analyzing packets. Wireshark supports a wide range of protocols and offers advanced features like filtering, coloring rules, and protocol dissection.
- **tcpdump:** A command-line packet analyzer that allows users to capture and display packets. tcpdump uses BPF syntax for filtering and provides options for detailed packet analysis.

- **Tshark:** The command-line version of Wireshark, offering similar functionality without the graphical interface. Tshark is useful for automated and script-based packet analysis.

3.2 Modern Approaches for Internet Traffic Classification

3.2.1 ML-Based Statistical Solutions

Statistical classification for network traffic combines statistical feature extraction with Machine Learning algorithms. The process begins by representing network traffic as flows, aggregating packets sharing 5-tuples: source and destination IP addresses, port numbers, and the Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) protocol. Next, statistical features are extracted at the packet or flow level. Packet-level features include packet length and inter-arrival time, derived from single or aggregate packets. Flow-level features, extracted from entire flows, include the total number of packets, total bytes, and flow duration. Machine Learning approaches in this context are categorized as supervised, unsupervised, or semi-supervised classifiers. In general, the machine learning approach follows the framework depicted in 3.3. The general process of the machine learning framework is summarized as follows:

- **Data Collection:** The initial step in network traffic classification involves gathering sufficient traffic data for targeted applications or protocols. This can be done through private data collection, which captures traffic from devices like computers and Internet of Things (IoT), or by utilizing public

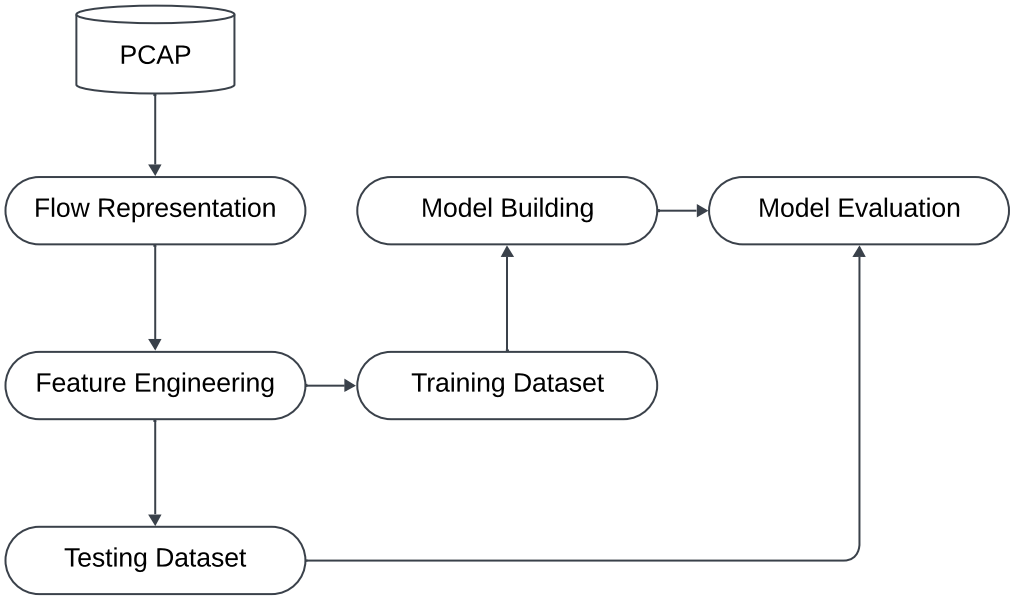

Figure 3.3: Supervised learning flowchart.

datasets provided by other researchers.

- **Flow representation:** Raw traffic is typically represented using flow aggregation, which groups packets sharing the same 5-tuple: source port, destination port, source IP, destination IP, and protocol (TCP or UDP). Flows can be unidirectional, aggregating packets in one direction, or bidirectional, covering both directions.

- **Feature engineering:** This process is vital for optimizing classifier performance, as it computes metrics reflecting traffic properties. Feature engineering typically involves feature extraction and reduction. Feature extraction identifies statistical features characterizing application flows, aiding in differentiation. It can be manual or automated, producing a 2D matrix where rows represent flows and columns represent features. Feature

reduction, though optional, removes irrelevant or redundant features, enhancing accuracy and efficiency. Common algorithms include wrappers, which iteratively assess subsets, and filters, which analyze datasets directly. Correlation-based Feature Selection (CFS) Hall (1999) uses relevance and redundancy measures to refine features, while Consistency-based Subset Search (CON) Dash et al. (2000) and Gain Ratio (GR) algorithms rank features by class relevance. GR, based on Information Gain (IG), and Chi-square Liu & Setiono (1995) methods further rank features using entropy or statistical measures.

- **Datasets preparation:** After extracting features and applying feature selection, the dataset containing historical data is ready for building and testing the classifier using either separate training and testing datasets or N-fold cross-validation. Separate datasets require a large training set for optimal training and an adequate testing set for accurate evaluation. Using the same dataset for both training and testing is poor practice as it can produce misleading results. N-fold cross-validation divides the dataset into N approximately equal folds, using $N-1$ folds for training and 1 fold for testing, iterating the process N times. Literature suggests that $N = 10$ provides acceptable classification performance Witten & Frank (2002).

- **Model building:** The training dataset from the previous step is used to build a model that classifies network flows by their causal applications/protocols. Various Machine Learning algorithms address tasks like classification and clustering, with the choice depending on the analyst's objectives. Two main groups, supervised and unsupervised algorithms, are

commonly applied, while semi-supervised approaches are gaining popularity.

- **Model evaluation:** Before deploying the solution into production, an evaluation analysis is required to assess the classifier's performance. In essence, performance metrics are calculated by comparing the model's predictions to the ground truth labels, tailored to the classification goal binary or multiclass. Binary classification involves assigning inputs to one of two classes, while multiclass classification involves choosing from multiple classes. Five common metrics for evaluating supervised classification models are accuracy, precision, recall, F-measure, and Receiver Operating Characteristic (ROC) curve Christen & Christen (2012). Accuracy measures the proportion of correctly classified instances but may be misleading for imbalanced datasets Azab et al. (2014). Precision, recall, and F-measure provide more robust insights. Precision assesses the accuracy of predictions for a class, recall evaluates how well the model captures instances of the target class, and F-measure is their harmonic mean. The equations for these metrics are outlined below.

Supervised machine learning algorithms generate knowledge structures to classify untrained instances into predefined classes Reich & Fenves (1991). The process involves two main phases: training and testing. During training, the algorithm is provided with labeled instances and their statistical features to build a classification model by extracting patterns that distinguish classes. The resulting knowledge may be represented as rules, a decision tree, or a flowchart, depending on the algorithm. In the testing phase, the classifier is used to cat-

egorize unseen instances. These algorithms are commonly applied to classify network traffic.

Williams et al. (2006) conducted a comparative analysis of various learning algorithms, including Bayes Net, C4.5, AdaBoost C4.5, NBTree, AdaBoost NBD, and Nearest Neighbor, for network classification. They also examined the Cybersecurity Framework (CSF), wrapper, and consistency feature selection algorithms. The study utilized the National Laboratory for Applied Network Research (NLANR) dataset NLANR (2020), comprising network traffic data from protocols such as Hypertext Transfer Protocol and Simple Mail Transfer Protocol. Key features like packet length and inter-arrival statistical values were extracted. Among the approaches, the wrapper feature selection algorithm combined with AdaBoost C4.5 delivered the highest accuracy. Additionally, they evaluated the speed of the building and testing phases, finding Naive Bayes using both kernel density estimation (NBK) to be the fastest in model building and C4.5 the quickest during testing.

Further research Fan & Liu (2017), Yuan et al. (2010) explored the performance of different Support Vector Machine (SVM) kernel functions (linear, polynomial, sigmoid, and radial) for network flow classification. A sequential forward feature selection algorithm was employed to eliminate irrelevant and redundant features, enhancing accuracy. The radial kernel emerged as the top performer among the kernels.

Jenefa & Moses (2018) compared the accuracy of C5.0 with C4.5, SVM, and Naive Bayes algorithms using features such as packet rate, data rate, and inter-arrival time statistics (minimum, mean, maximum, and standard deviation). Their experiments, conducted on a private dataset of 17 applications collected

in their lab, demonstrated the superiority of C5.0, achieving 99% recall and precision for most classes.

Dias et al. (2019) proposed a machine learning framework for classifying real-time video network traffic, such as YouTube and Netflix streams. The framework was trained on 13 statistical features, including packet arrival time, average decimal values, and IP datagram length. The dataset was collected in their lab, and the proposed algorithm achieved an average accuracy of 98.88%, outperforming Naive Bayes in both setup and classification speed, thus demonstrating its effectiveness for real-time application classification.

Alshammari & Zincir-Heywood (2010) explored the effectiveness of three machine learning algorithms; AdaBoost, C4.5, and Genetic Programming (GP) for classifying network flows. Their study utilized statistical features, such as flow duration and packet length and arrival interval statistics in both directions. The dataset, privately collected, included VoIP traffic (e.g., Skype, Gtalk, and Yahoo Messenger) and non-VoIP traffic. C4.5 outperformed the other algorithms, achieving a 99% Detection Rate (DR) for Skype and Gtalk traffic classification with False Positive Rates (FPR) below 1% and 0.2%, respectively. They also examined the robustness of the classifier Alshammari & Zincir-Heywood (2009), comparing AdaBoost, SVM, NB, RIPPER, and C4.5 using four datasets: a privately collected dataset, the NLANR dataset NLANR (2020), the MAWI dataset MAWI Working Group et al. (2012), and the DARPA dataset Thomas et al. (2008). C4.5 consistently demonstrated superior accuracy for detecting Skype and SSH traffic.

Sun et al. (2018) introduced the Incremental SVM (ISVM) to reduce the computational demands of traditional SVM classifiers, addressing the high train-

ing cost in terms of memory and CPU. They also proposed the Authenticator ISVM (AISVM) framework, leveraging data from previous training sessions. Their experiments showed that ISVM and AISVM offered higher accuracy with lower computational requirements compared to standard SVM. Naive Bayes and Naive Bayes with kernel density estimation (NBKDE) algorithms were tested but yielded lower accuracy than SVM.

Cao et al. (2020) aimed to enhance the traffic classification accuracy of SVM-based models through a two-module approach: feature engineering and classifier building. The feature engineering module implemented a filter-wrapper mixed feature selection algorithm to identify the most representative features, while the classifier building module used an Improved Grid Search optimization algorithm to fine-tune SVM parameters. Using the Moore & Zuev (2005) dataset, their proposed solution demonstrated high classification accuracy with a reduced feature set, outperforming SVM, NB, and kNN algorithms.

Khatouni & Zincir-Heywood (2019) conducted a comparative analysis of different network traffic flow exporters named Bullard (2005), Sanders et al. (2020), Finamore et al. (2011), and Burschka & Dupasquier (2016) for classifying traffic. They evaluated their models using the NIMS dataset Shahbar & Zincir-Heywood (2017) collected from various locations, focusing on robustness across diverse service traffic, including browsing and audio flows. The results highlighted the model's robustness, achieving an average True Positive score of 85% when tested across different locations.

Dong (2021) proposed an enhanced SVM method called cost-sensitive SVM (CMSVM) to improve accuracy, address computational costs, and resolve data imbalance issues. This approach incorporates an active learning technique to dy-

namically assign weights for specific applications. The CMSVM was evaluated using the Moore & Zuev (2005) datasets to classify network flows into corresponding service groups. Results demonstrated that CMSVM outperformed the traditional SVM classifier in terms of accuracy and handling imbalanced data.

Afuwape et al. (2021) explored the classification of VPN and non-VPN network traffic using ensemble classifiers, focusing on precision, recall, and F1-score. Their experiments, conducted with the ISCXVPN (2016) dataset, compared the performance of various ensemble and single classifiers. Gradient Boosting (GB) and Random Forest (RF) ensemble classifiers showed superior accuracy compared to single classifiers such as decision tree, Multi-Layer Perceptron (MLP), and kNN.

Ganesan et al. (2021) proposed a machine learning-based scheduling framework designed to prioritize network traffic in IoT environments based on Quality of Service (QoS) requirements. Their study compared the performance of seven supervised learning algorithms, including RF, kNN, MLP, Naive Bayes, logistic regression (LR), and SVM, using the Sivanathan et al. (2018) dataset with data from 21 IoT and non-IoT devices. Random Forest achieved the highest accuracy among the evaluated models.

The studies summarized in Table 3.2 demonstrate a wide range of supervised machine learning approaches applied to network traffic classification. The results highlight the importance of selecting appropriate algorithms, feature selection methods, and datasets for achieving high classification accuracy. Techniques such as AdaBoost, SVM (including its variants like CMSVM and ISVM), C4.5, C5.0, Naive Bayes, and ensemble methods like Random Forest and Gradient Boosting have been widely used across diverse datasets and applications.

Study	Multi Class	Low-Latency	Granularity	Algorithms	Dataset	Focus	Accuracy
Williams et al. (2006)	6 Classes	No	Protocol	Bayes Net, C4.5, NBTree, SVM	NILMA	Accuracy, Computational Time	94.4
Fan & Liu (2017)	8,7 Classes	Yes	Service Group	SVM	Moore, Private	Accuracy	97.17
Jenefa & Moses (2018)	17 Classes	Yes	Application	C5.0, C4.5, SVM, NB	Private	Accuracy	98.07
Dias et al. (2019)	3 Classes	No	Application	NB, Novel framework based on NB	Private	Accuracy, Computational Time	98.88
Alshammari & Zincir-Heywood (2010)	Binary	Yes	Application	AdaBoost, C4.5, GP	Private	Accuracy	99
Alshammari & Zincir-Heywood (2009)	Binary	Yes	Application	AdaBoost, SVM, NB	Private	Accuracy	98.4
Sun et al. (2018)	10 Classes	No	Service Group	NB, NBKDE, SVM, ISVM	Moore	Accuracy, Computational Time	96
Cao et al. (2020)	10 Classes	No	Service Group	SVM, improved SVM, NB	Moore	Accuracy	99.34
Khatouni & Zincir-Heywood (2019)	11 Classes	No	Service Group	Decision Tree	NIMS	Accuracy, Robustness	85
Dong (2021)	10 Classes	Yes	Service Group	SVM,CMSVM	Moore	Accuracy, Computational Cost	94
Afuwape et al. (2021)	7 Classes	Yes	Service Group	GB, RF, kNN, Decision Tree	ISCX	Accuracy	93.8
Ganesan et al. (2021)	21 Classes	No	Application	RF, kNN, MLP, NB, LR	UNSW	Accuracy	99

Table 3.2: Summary of ML-based statistical solutions.

High accuracy was consistently achieved by approaches incorporating advanced feature selection, optimization, or tailored frameworks, such as C5.0 (Jenefa & Moses (2018)), Improved SVM (Cao et al. (2020)), and CMSVM (Dong (2021)). Applications with higher class granularity or real-time demands often favored ensemble or optimized algorithms, as seen with Random Forest for IoT traffic (Ganesan et al. (2021)) and the framework by Dias et al. (2019) for video traffic. Collectively, these findings underline the ongoing advancements in ML-based statistical solutions for robust and efficient network traffic classification.

3.2.2 Deep Learning Solutions

Neural Networks (NN) are systems of interconnected neurons that process information based on their response to inputs. Weights on the connections are adjusted using methods like backpropagation. Deep learning, a variant of NN with many hidden layers, commonly uses models such as Multilayer Perceptron, Recurrent Neural Network, Autoencoders, and Convolutional Neural Network, particularly for tasks like network traffic classification.

- **Multilayer Perceptron**: A feedforward NN with input, hidden, and output layers, where each neuron applies a non-linear activation function. MLP is complex and hard to train due to numerous parameters, making it more suitable as part of a framework rather than a standalone solution.

- **Recurrent Neural Network**: Designed to capture temporal correlations, RNN uses feedback connections between layers for sequential data processing. It excels in tasks like speech recognition and time-series anomaly detection. LSTM, a variant of RNN, addresses the vanishing gradient prob-

lem to enable long-term dependencies.

- **Convolutional Neural Network**: Specialized for spatial data like images, CNN processes inputs through filters that slide over pixel values to extract features. Features are pooled, flattened, and passed to fully connected layers for classification, making CNN highly effective in image recognition.

- **Auto Encoder**: An unsupervised model that reconstructs inputs by compressing data with an encoder and reducing dimensionality. AE is used for feature extraction and initialization in deep learning. Stacked Auto-Encoders (SAEs) extend AE by stacking layers, employing a greedy layer-wise training approach.

Recently, researchers have integrated deep learning to classify network traffic into application or service groups. Using raw network traffic as input, deep learning models identify spatial or temporal patterns without requiring feature engineering, making them more accessible and convenient to implement. CNN, a type of deep neural network, excels in identifying spatial correlations and has proven effective in fields like image recognition. CNN simplifies images for processing while preserving essential features for accurate predictions. It represents the input image as a vector of pixel values and applies a filter that slides across the image with a defined stride, extracting features across its width. The filter then moves down and repeats the process until the entire image is covered. The number of features extracted depends on parameter tuning. The resulting feature map is passed to a pooling layer, which reduces its spatial size. Finally, the output is flattened and fed into a fully connected neural network for classification.

The use of deep learning classifiers for network traffic classification has recently gained popularity, eliminating the need for manual feature engineering. Researchers have used CNNs to capture spatial correlations and RNNs for temporal correlations in traffic data. Models are trained using raw traffic data or extracted statistical features from collected flows.

Wang (2015) was among the pioneers in applying deep learning techniques to classify network traffic. The proposed framework utilized Artificial Neural Networks (ANN) and Stacked Auto Encoders (SAE) to categorize traffic based on their associated applications. The dataset, privately collected, included over 58 encrypted and unencrypted protocols. Evaluation results demonstrated the solution's efficiency and emphasized the critical role of the first n bytes in the classification process.

Wang, Zhu, Zeng, Ye & Sheng (2017) explored the use of CNN-2D to classify raw network traffic by leveraging spatial correlations. Traffic data were grouped into unidirectional and bidirectional flows, with the initial 784 bytes of each flow used for early-stage identification. Flows shorter than 784 bytes were zero-padded. Each flow was converted into 28x28 pixel images to serve as input for the CNN classifier. Using the USTC-TFC2016 dataset yung (2016), which includes both malicious and benign traffic for various applications, the approach effectively distinguished between these traffic types.

Wang, Zhu, Wang, Zeng & Yang (2017) employed CNN-1D to classify network traffic, independent of whether it originated from VPN tunnels. The authors argued that CNN-1D is better suited for processing sequential data like network traffic. The ISCX dataset ISCXVPN (2016) , containing both VPN and non-VPN traffic across applications, was used for evaluation. Results indicated

that CNN-1D achieved higher accuracy compared to CNN-2D and C4.5 when analyzing 784 bytes.

Lopez-Martin et al. (2017) integrated spatial and temporal characteristics for traffic classification by combining CNN for spatial correlations and LSTM for temporal correlations. For image preparation, the first 20 packets of each flow were used, extracting six features from each packet: source port, destination port, payload size, TCP window size, inter-arrival time, and direction. Using the RedIRIS dataset, the combined CNN-LSTM model delivered superior classification results.

Lotfollahi et al. (2020) proposed a deep packet framework combining CNN and Stacked Auto Encoder (SAE) for network traffic classification. The SAE architecture included five fully connected layers, while the CNN model consisted of two CNN-1D layers. They observed that most packets had a payload size of 1480 bytes, which was used in their experiments. Evaluations on the ISCX dataset ISCXVPN (2016) showed that their framework was effective in both application identification and service group classification.

Huang et al. (2018) developed a Multi-Task Learning architecture based on CNN for network traffic classification. The architecture included three convolutional layers and achieved high accuracy. Using the CTU-13 and ISCX datasets ISCXVPN (2016), the framework effectively classified malicious and benign traffic into their respective classes, utilizing 1024 bytes (represented as 32x32 pixel images).

Wang et al. (2018) proposed the SDN-HGW framework to manage Quality of Service (QoS) in smart home networks. This framework incorporates multilayer perceptrons, stacked autoencoders (SAE), and convolutional neural networks

(CNN) using the ISCX dataset ISCXVPN (2016), which includes data traffic from 15 applications. Each application's traffic was represented as a 1480-byte vector input to the framework. Experimental results demonstrated high accuracy with minimal computational overhead, enabling real-time decision-making.

Aceto et al. (2019*b*), Aceto et al. (2018) compared the performance of various deep learning models for mobile network traffic classification. They used a privately collected, unbiased dataset to conduct binary and multiclass classification evaluations. Their results showed that CNN-1D, using a 784-byte input, delivered the highest accuracy.

Zhou et al. (2017) applied min-max normalization to convert network traffic flow features into grayscale images, which were then used as input for CNN classifiers. By transforming Moore's 249 statistical features into 16×16 pixel images, they achieved a highly accurate classifier for categorizing network flows by service groups.

Tong et al. (2018) combined a CNN classifier with a supervised learning algorithm to classify QUIC-based network traffic. Initially, flow statistical features were extracted to distinguish Google Hangouts from other applications. Packet-level features were then transformed into images and analyzed by CNN-1D to classify flows into their respective services. Accuracy evaluations on a privately collected dataset demonstrated the framework's effectiveness.

Chen et al. (2017) introduced a framework called seq2image, which converts network flows into images using RKHS kernel embedding, based on just 10 packets. When evaluated on a private dataset containing five protocols, the framework outperformed traditional supervised learning algorithms like Naive Bayes (NB), Support Vector Machines (SVM), and decision trees.

Aceto et al. (2019*a*) proposed the Multimodal DL-based Mobile Traffic Classification (MIMETIC) framework, designed to leverage the diverse nature of mobile network traffic by learning both intra-modality and inter-modality dependencies. This approach addressed the limitations of single-modality methods by analyzing traffic patterns from two perspectives, employing CNN for payload features and Gated Recurrent Unit (GRU) for time-series data. To evaluate their solution, the authors collected private datasets of traffic from iOS and Android applications. Their model achieved an accuracy of 89.49% and 89.14% for Android and iOS traffic, respectively, by analyzing 576 bytes and 12 packets per flow.

In a related study, the same authors Aceto et al. (2020) presented a taxonomy comparing traditional statistical machine learning and deep learning classifiers, emphasizing the advantages of deep learning methods. They also introduced a generalized deep learning framework for classifying mobile network traffic for both iOS and Android applications. This framework offered flexibility by allowing the use of CNN or RNN algorithms and supporting single-modal or multi-modal inputs. It processed the first N bytes or packets while analyzing packet payloads or headers. The evaluation on a custom human traffic dataset demonstrated the framework's superior performance compared to baseline deep learning and machine learning classifiers.

Bu et al. (2020) proposed a deep learning approach for network traffic classification using a deep parallel Network-In-Network (NIN) architecture with multiple MLP convolutional modules. This framework employs a parallel decision strategy by constructing two sub-networks that separately process the packet payload and packet header. Unlike traditional CNNs, NIN integrates a micro-

network after each convolution and uses global average pooling before the final classification stage, reducing the number of model parameters. When tested on the ISCX dataset ISCXVPN (2016), the proposed method achieved higher accuracy than traditional CNNs by analyzing 1480 bytes of the monitored flow.

Similarly, Liu et al. (2019) developed an end-to-end framework called Flow Sequence Network (FS-Net) for network traffic classification. This model identifies features directly from raw traffic data, eliminating the need for manual feature engineering. FS-Net consists of three main components: an encoder for feature extraction, a decoder for input sequence reconstruction, and a softmax classifier for network flow categorization. The framework was evaluated using a dataset collected by Liu et al. (2018), which included traffic from various applications. The results demonstrated that FS-Net outperformed other state-of-the-art methods in classifying network traffic.

The table 3.3 summarizes various deep learning solutions for network traffic classification, highlighting traffic types, algorithms, datasets, and achieved accuracies. Studies predominantly focus on raw traffic or packet-level features, leveraging CNNs, RNNs, and hybrid models like CNN-LSTM for application or service group classification. Accuracy rates generally exceed 95%, showcasing the efficiency of deep learning approaches. Key datasets include ISCX, USTC-TFC2016, and privately collected data, with methodologies ranging from feature extraction to direct raw data processing. Some studies specifically address encrypted traffic, VPN/non-VPN differentiation, and malicious/benign classification, demonstrating deep learning's adaptability to diverse traffic scenarios and granularities.

The dataset table 3.4 provides a comprehensive overview of used datasets

Study	Traffic Type	Low-Latency	Granularity	Algorithms	Input	Dataset	Accuracy
Wang (2015)	Encrypted/ Non-Encrypted	No	Application	SAE	Raw Traffic	Private	95
Wang, Zhu, Zeng, Ye & Sheng (2017)	Malicious/ Benign	Yes	Application	CNN	Raw Traffic	USTC-TFC2016	99.41
Wang, Zhu, Wang, Zeng & Yang (2017)	VPN/Non-VPN	Yes	Service Group	CNN	Raw Traffic	ISCX	80 to 99
Lopez-Martin et al. (2017)	Mix App Traffic	No	Application	CNN, LSTM	Packet Level Feature	RedIRIS	96.5
Lotfollahi et al. (2020)	VPN/Non-VPN	Yes	Application/ Service Group	CNN, SAE	Raw Traffic	ISCX	98
Huang et al. (2018)	Malware/ VPN/Non-VPN	No	Application	CNN	Raw Traffic	CTU-13/ ISCX	98.4
Wang et al. (2018)	VPN/Non-VPN	Yes	Application	CNN, MLP, SAE	Raw Traffic	Yes	99
Aceto et al. (2019*b*)	Android/ IOS App Traffic	No	Application	CNN, SAE,LSTM	Raw/ Packet Level Feature	Private	96.8
Zhou et al. (2017)	App to Service Group	Yes	Service Group	CNN	Flow Statistical Features	Moore	98.6
Tong et al. (2018)	QUIC Traffic	Yes	Application	RF, CNN	Flow/ Packet Features	Private	97
Chen et al. (2017)	Various Protocols	No	Protocol	CNN	Raw Traffic	Private	88.42
Aceto et al. (2019*a*)	Android/ IOS App Traffic	No	Application	CNN, GRU	Raw Traffic/ Time Series Features	Private	96.74
Aceto et al. (2020)	Android/ IOS App Traffic	No	Application	CNN, RNN	Raw Traffic/ Time Series Features	Private	90 to 99
Bu et al. (2020)	VPN/Non-VPN	Yes	Application	MLP	Raw Traffic	ISCX	99
Liu et al. (2019)	Mix App Traffic	No	Application	RNN, AE	Raw Traffic	Private	96

Table 3.3: Summary of deep learning solutions.

for network traffic classification research, detailing their characteristics and formats. Each dataset is identified by the number of traffic classes, traffic types, labeling status, and year of publication. Notable datasets include ISCX, USTC-TFC2016, and CTU-13, which provide labeled traffic data in PCAP format, enabling deep learning models to classify applications, services, or protocols. The Moore and NIMS datasets offer ARFF formats for specific traffic categories, while the MAWI dataset spans two decades but lacks labeled data, making it suitable for unsupervised learning. These datasets underpin the evaluation of deep learning models, ensuring their robustness across diverse traffic types and scenarios.

Dataset	# of Classes	Traffic Types	Year	Labelled	Format
Moore	10	VPNWeb, SMTO, POP3, FTP, DNS, BitTorrent, MySQL, Virus, Windows media player, Telnet, WOW	2005	Yes	ARFF
ISCX	7	HTTPS, SMTP, Facebook, Chrome, FTP, Skype, BitTorrent	2016	Yes	Full PCAP
USTC-TFC2016	2	Facetime, Gmail, Skype, Zeus, Cridex, Htbot	2016	Yes	Full PCAP
CTU-13	13	Botnet sample for each scenario	2014	Yes	Full PCAP/ ARFF
MAWI	Varies	HTTP, FTP, SSH, DNS, SSL, DoS, other	2000-2020	No	Full PCAP
NIMS	Varies	Browsing, IRC, streaming, Torrent, other	2014-2017	Yes	ARFF

Table 3.4: Dataset summary

3.2.3 Wavelet Transform in Classification

The application of wavelet transform in network traffic classification has gained significant attention in recent years due to its ability to handle non-stationary time series data effectively. This section reviews notable contributions to the field, highlighting methodologies and findings from key studies.

Girma et al. (2000) introduces a novel method for online video traffic modeling using wavelet transform techniques. The research focuses on leveraging wavelet transforms to analyze and simplify video traffic data. A key aspect of the proposed model is the removal of less significant wavelet components, which optimizes time resolution and reduces computational overhead. This method enables efficient online modeling by estimating model parameters through prior analysis of the autocorrelation function of base video traffic traces. The study demonstrates that the wavelet-based approach is effective for capturing the essential dynamics of video traffic while maintaining a manageable complexity for real-time applications.

Tan et al. (2013) explores an innovative approach to predicting network traffic using wavelet transform techniques. The authors propose a model that decomposes complex and non-stationary traffic signals into multiple frequency components, enabling precise temporal and frequency-domain analysis. The algorithm leverages wavelet-based decomposition to extract patterns from both high-frequency (short-term variations) and low-frequency (long-term trends) components, which are then used to forecast traffic. The study's findings highlight the algorithm's ability to adapt to real-world network traffic dynamics, outperforming traditional prediction methods in accuracy and robustness. This

improved accuracy stems from the model's capacity to handle nonlinearities and noise within traffic data. The results emphasize the algorithm's potential applications in network resource optimization, traffic shaping, and proactive congestion management, offering significant improvements for network stability and efficiency.

Shi et al. (2017) explores a comprehensive study aimed at enhancing the efficiency and robustness of traffic classification in network systems. The authors propose an integrated framework that combines feature extraction, selection, and optimization techniques to address challenges such as high-dimensional data, computational complexity, and classification accuracy. By leveraging advanced statistical methods and machine learning algorithms, the framework identifies critical features that contribute significantly to classification tasks while discarding irrelevant or redundant ones. Key findings include a significant reduction in computational overhead without compromising classification accuracy. The authors tested their approach on real-world datasets and demonstrated improved performance compared to traditional methods. For instance, their proposed method achieved higher classification accuracy (over 95%) on several datasets while reducing the feature set size by up to 60%, showcasing its efficiency. Additionally, the framework proved robust under varying traffic scenarios and noise levels, ensuring consistent performance in practical deployments. These results underline the potential of the proposed method to enhance traffic classification in modern network systems.

long Fei et al. (2017) presents an innovative approach to traffic classification by addressing challenges inherent in supervised machine learning, such as the need for labeled samples and the inability to identify unknown traffic types.

The authors combine wavelet transform and an improved K-means clustering algorithm, leveraging the wavelet transform's strength in handling multi-fractal network traffic. The proposed method uses the wavelet spectrum of scatter factors to represent stream sequences, which then serve as input for the clustering algorithm. The improved K-means algorithm incorporates genetic algorithms (GA) to enhance clustering stability and accuracy. Experimental results demonstrate that this method outperforms conventional approaches in both stability and classification accuracy, making it a robust tool for identifying diverse traffic types in network systems.

Ertam & Avcı (2017) presents a novel framework combining Genetic Algorithm (GA) optimization and Wavelet Kernel Extreme Learning Machines (WK-ELM) for classifying internet traffic. Recognizing the growing importance of traffic classification in network management, the study employs machine learning techniques to address challenges in accuracy and computational efficiency. The methodology uses a GA-based optimization process to select the most effective parameters for the WK-ELM classifier, where the wavelet function enhances feature representation. This hybrid approach achieves a classification accuracy exceeding 95%, outperforming traditional methods. Evaluation metrics include the average truth value for performance comparison and Receiver Operating Characteristic (ROC) curves, demonstrating the model's superior predictive capability. This innovative combination of GA and WK-ELM not only highlights the potential of hybrid methods in machine learning but also establishes a reliable framework for real-world applications in network traffic analysis and management. The results emphasize the efficiency of using GA for feature selection and optimization within complex machine learning algorithms.

Liang et al. (2019) presents a novel approach to enhancing the recognition of network traffic content by integrating wavelet transform and convolutional neural networks (CNNs). The authors demonstrate that the wavelet transform effectively extracts time-frequency domain features, enabling the analysis of complex patterns within network traffic, even when the data is encrypted or otherwise obfuscated. The CNN component of the model leverages these features to perform robust and accurate classification of traffic content. The findings highlight the method's superior performance compared to traditional techniques, particularly in terms of recognition accuracy and adaptability to diverse network conditions. The study further underscores the practical value of this hybrid approach in real-world applications such as intrusion detection, traffic analysis, and quality of service optimization. The results emphasize that combining time-frequency analysis with deep learning enhances the capability of systems to address challenges in modern network environments.

Dillbary et al. (2024) introduces two novel methods, STFT-TC (Short-Time Fourier Transform Traffic Classification) and DWT-TC (Discrete Wavelet Transform Traffic Classification), to address challenges posed by encrypted traffic. These methods use compact time-series representations coupled with advanced digital signal processing techniques. The STFT-TC method focuses on extracting statistical and spectral features from magnitude spectrograms, offering innovative insights for encrypted traffic classification. In contrast, the DWT-TC method leverages wavelet coefficients to extract statistical features, capturing the signal's shape and energy distribution. The researchers evaluated their methods using two public QUIC datasets, showing significant improvements. Their models achieved up to 5.7% higher accuracy and up to 5.9% better F1-scores, show-

casing the potential of these spectral and multiresolution approaches to enhance traffic classification in an era of encrypted communication.

In summary, these studies collectively demonstrate the versatility of wavelet transform in network traffic classification and prediction. The techniques reviewed here showcase advancements in accuracy, computational efficiency, and robustness, as summarized in Table 3.5. This highlights the importance of wavelet transform in addressing contemporary challenges in network traffic analysis.

Study	Year	Methodology	Key Findings
Girma et al. (2000)	2000	Wavelet-based model for online video traffic prediction	Efficient parameter estimation for real-time traffic modeling.
Tan et al. (2013)	2013	Multi-resolution analysis with ARMA Model	Enhanced prediction accuracy for short-range traffic.
Shi et al. (2017)	2017	WLMF feature extraction and PCA	Improved classification accuracy and computational efficiency.
long Fei et al. (2017)	2017	Wavelet spectrum with improved K-means clustering	Stability and accuracy improvements in unsupervised traffic classification.
Ertam & Avcı (2017)	2017	GA-WK-ELM framework with wavelet functions	Over 95% classification accuracy with computational efficiency for real-time classification.
Liang et al. (2019)	2019	Wavelet transform with CNN	High accuracy in content-based traffic classification.
Dillbary et al. (2024)	2024	DWT with statistical feature extraction	Up to 5.7% improvement in classification accuracy for encrypted traffic analysis.

Table 3.5: Key studies on wavelet transform in network traffic classification

3.2.4 Advanced Trend Features for Internet Traffic Classification

The progression of research on feature selection and extraction for internet traffic classification reveals significant advancements over the years. Liu & Liu (2012) introduced the Balanced Feature Selection (BFS) method, focusing on addressing the class imbalance problem in traffic classification by measuring features both locally and globally. Their BFS approach used a certainty coefficient to measure correlations and optimize feature subsets, offering an innovative filter-based solution for achieving balanced classification performance. In the same year, Zhang et al. (2012) proposed a hybrid feature selection approach that combined Weighted Symmetrical Uncertainty (WSU) and Area Under the Curve (AUC) metrics to tackle the imbalance in traffic flows. Their method effectively improved classification accuracy by selecting robust and stable features using dynamic traffic flows, achieving more than 94% flow accuracy and 80% byte accuracy across diverse datasets.

Building on these foundations, Shi et al. (2017) presented a novel approach combining Wavelet Leaders Multifractal Formalism (WLMF) and Principal Component Analysis (PCA) for feature extraction and selection. Their technique addressed limitations in traditional methods by capturing complex non-linear traffic characteristics and improving machine learning classifiers' robustness. The experimental results demonstrated significant accuracy improvements in real-time traffic classification using Support Vector Machines (SVMs). Most recently, Adje et al. (2022) leveraged interpretable machine learning models such as decision trees and XGBoost to identify discriminating features for traffic clas-

sification. Their approach utilized reduced feature sets and achieved exceptional classification accuracy (99.76% on one dataset and 98.40% on another). Their work also highlighted the significance of specific features, such as port number and minimum segment size, which contributed to the efficiency and stability of their models.

This chronological advancement underscores continuous refinement of techniques, from balancing feature selection to developing robust extraction methods and leveraging interpretable models, addressing challenges like imbalanced datasets, real-time classification, and feature significance in internet traffic classification.

3.2.5 Alternative Approaches in Internet Traffic Classification

Apart from the above methodologies, other Internet traffic analysis techniques have also drawn interest. Early studies, such as Crotti et al. (2007), leveraged simple statistical fingerprinting to classify traffic flows based on properties like packet size and inter-arrival time, introducing the concept of "protocol fingerprints" for compact and efficient classification. Wang & Parish (2010) advanced this by optimizing multi-stage classifiers tailored for TCP traffic, focusing on packet size distributions to enhance detection accuracy. As encrypted and proprietary traffic increased, Qin et al. (2015) tackled the challenge by introducing robust methods for identifying P2P and VoIP applications using Packet Size Distribution (PSD) signatures and Bi-flow models, demonstrating resilience to packet loss.

With machine learning gaining prominence, Schmidt et al. (2017) optimized artificial immune systems (AIS) for flow-based classification, improving effi-

ciency and accuracy while addressing parameter sensitivity. Around the same time, Lashkari et al. (2017) and Draper-Gil et al. (2016) investigated time-based features to characterize encrypted and VPN traffic, providing valuable datasets and frameworks for identifying traffic flows, including Tor. These efforts laid the groundwork for advanced algorithms capable of tackling encrypted communication.

Modern advancements introduced deep learning to traffic classification. Lotfollahi et al. (2020) proposed the "Deep Packet" framework, integrating feature extraction and classification using CNNs, achieving high accuracy in distinguishing VPN and non-VPN traffic. Similarly, Shapira & Shavitt (2019) introduced "FlowPic," transforming flow data into images and applying CNNs to classify encrypted traffic with remarkable precision. These innovations built upon earlier approaches by unifying feature extraction and classification, thus overcoming limitations of manually crafted features.

Together, these studies illustrate a continuous refinement of techniques, from statistical methods to machine learning and deep learning, each addressing challenges posed by encryption, high-speed networks, and evolving traffic patterns. This trajectory highlights the adaptability and innovation driving internet traffic classification.

3.3 Summary and Research Gaps

Advancements in internet traffic classification highlight the evolution from traditional methods like deep packet inspection (DPI) to modern machine learning (ML) and deep learning (DL) solutions. While ML and DL have improved

accuracy and reduced dependency on manual feature engineering, challenges remain in handling encrypted traffic, achieving real-time classification, and ensuring scalability in high-speed networks. Wavelet-based techniques have added value by analyzing time-frequency characteristics but lack broad application in dynamic environments.

Despite progress, significant research gaps persist. A notable limitation is the lack of emphasis on integrating trend-based features to capture evolving traffic patterns. Additionally, many approaches fail to address dynamic and real-time traffic identification in mixed and complex environments. Few studies analyze both time and frequency domain characteristics of network traffic comprehensively, limiting their ability to adapt to emerging protocols and encrypted communication. Furthermore, computational efficiency and interpretability remain critical challenges for deploying models in real-world scenarios.

Addressing these gaps will enable more robust, adaptable, and efficient traffic classification systems capable of meeting the demands of modern networks.

CHAPTER 4

Low-Latency Internet Traffic Identification using Machine Learning with Trend-based Features

This chapter presents a methodology for low-latency internet traffic classification using Machine Learning models trained with novel trend-based features. The key contribution of this chapter is the introduction of advanced trend analysis techniques that capture the underlying periodicity and long-term patterns in network traffic, which have not been sufficiently explored in previous work. Furthermore, the integration of these trend-based features with machine learning algorithms—such as k-NN, SVM, and Random Forest—demonstrates significant improvements in classification accuracy, particularly for low-latency traffic in mixed environments. This approach marks a departure from traditional methods by focusing on the time-varying characteristics of traffic, providing a more robust and adaptive solution for real-time traffic identification.

4.1 Motivation

As the demand for real-time applications grows, such as video conferencing, online gaming, and live financial trading, the need for accurate and efficient low-latency Internet traffic classification becomes critical. Traditional classification methods, as discussed in previous sections, often struggle in environments with encrypted traffic, mixed traffic types, or dynamic network conditions. These challenges are exacerbated by the widespread adoption of encryption protocols like HTTPS, which obscure payload content, and the use of dynamic port assignments, which disrupt traditional port-based classification techniques. While metadata does provide valuable information, it often lacks the granularity required to discern specific traffic types accurately, particularly when encryption and obfuscation limit the visibility of deeper packet-level details.

Despite metadata's utility, it primarily focuses on high-level attributes such as IP addresses, port numbers, and packet sizes, which are not always sufficient for capturing intricate traffic patterns. For instance, encrypted traffic from multiple applications may share similar metadata attributes but differ significantly in temporal and behavioral characteristics, making traditional methods prone to inaccuracies. These limitations necessitate advanced techniques capable of extracting meaningful patterns from dynamic and encrypted network environments.

This chapter introduces a novel approach to low-latency Internet traffic classification by incorporating trend-based feature extraction techniques, which capture long-term patterns and periodicity in network traffic. Specifically, five new trend-based features are introduced, each designed to capture the temporal

characteristics of network traffic more effectively. These features address gaps left by traditional methods by focusing on the underlying behavioral trends that remain consistent, even in encrypted and obfuscated traffic.

Furthermore, the use of wavelet transforms enhances the ability to analyze traffic by providing a detailed time-frequency representation. Unlike static methods, wavelet transforms allow for localized analysis of transient and persistent features, making them particularly suited for identifying low-latency traffic characteristics. When integrated with machine learning (ML) algorithms, such as k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), and Random Forest (RF), this approach offers a more robust and adaptive solution for real-time traffic classification.

The motivation for using machine learning stems from its ability to adapt to complex, high-dimensional data. ML models, when trained on features derived from trend analysis and wavelet transforms, can effectively learn and generalize traffic patterns, even in heterogeneous and dynamic environments. The combination of these advanced feature extraction techniques with ML enables more precise classification of low-latency traffic, particularly in challenging scenarios involving mixed and encrypted traffic.

The novelty of this chapter lies in:

- The use of trend-based features to identify long-term patterns in traffic that are essential for distinguishing low-latency applications.
- A comprehensive evaluation of the performance of various machine learning algorithms, including k-NN, SVM, and Random Forest, when trained with these new features.

- The integration of wavelet transforms to capture transient and persistent traffic features, providing a nuanced understanding of time-frequency characteristics.
- The ability of this approach to enhance classification accuracy for encrypted and mixed traffic types, outperforming traditional models, especially for real-time, latency-sensitive applications.

4.2 Methodology

The proposed network traffic classification helps identify applications that require low latency from mixed traffic by using new features to recognize trends. In essence, the approach exhibits two main stages: (1) extraction of significant features from network traffic and (2) classification using ML algorithms. The use of two-step processing ensures that the method can identify the traffic demanding low latency in an effective manner, thus also improving the general performance of the method. Below, the methodology and process involved are further detailed, which includes data collection, feature selection, scaling of the data for pattern recognition, and presentation of experimental results.

4.2.1 Dataset

Data collection was carried out on a local WiFi network and the set of traffic traces was composed of file transfer, video streaming, video conferencing and a mixture of these. These traces were collected over a 100 Mbps Internet connection; the measurements of throughput of active traffic in either direction were taken and recorded. In total, we have slightly more than 400,000 samples of

throughput values (more than 20 hours of application usage), with more than 60,000 samples for each type of traffic. The classes and sample sizes used for each class are presented in Table 4.1.

Traffic Line	Application	Total Samples
VoIP	Skype, Zoom, MS Teams	62177
Video Streaming	YouTube, Netflix, Prime	68987
File Transfer	FTPS, Torrent, Steam	72366
Mix Traffic	All	214987

Table 4.1: Traffic types, applications and sample sizes

4.2.2 Features

After collecting samples of different Internet traffic, we first extracted different features from them. Table 4.2 shows the new dataset after feature extraction. We have 12 features in total, and 5 of them are novel trend-based features. We used these features while training the ML algorithms. The description of the features is given in Table 4.3. The average throughput calculation has been done for a time period of 3 seconds.

New Dataset with Features											
D	U	DU (A)	DA3 (B)	UA3 (C)	rDA3 (D)	rUA3 (E)	**Rank of A**	**Rank of B**	**Rank of C**	**Rank of D**	**Rank of E**
50350000	622020	74,92	38,318	0,45896	1,02230	1,02214	**9**	**9**	**8**	**8**	**8**
40970000	379980	107,82	37,482	0,44901	1,0060	1,00809	**9**	**9**	**8**	**6**	**6**
30120000	602670	49,97	37,256	0,44541	0,99404	1,027	**8**	**9**	**8**	**5**	**8**

Table 4.2: Dataset v2

ID	The Features
DU	(D)ownload to (U)pload ratio
DA3	The average throughput in downstream over 3 sec
UA3	The average throughput in upstream over 3 sec
rDA3	Throughput ratio (slope) in downstream
rUA3	Throughput ratio (slope) in upstream

Table 4.3: Novel trend features. "r" refers to ratio.

4.2.3 Slope

Experiments with traffic pattern analysis proved the importance of the changing throughput. A new feature based on the slope of a linear function, representing throughput changes, was developed. The slope is calculated as the ratio of vertical change between two points to horizontal change between the same two points. Figure 4.1 shows the file transfer traffic pattern on the left and the slope is calculated for the time interval 97-100 on the right. This new attribute contributes to the identification of a general trend in the throughput rate: A positive slope indicates an increasing rate; a negative slope indicates a decreasing rate. This feature was generalized from the different single and mixed compositions of Internet traffic to give the ability to generalize traffic patterns and view the data as a whole.

4.2.4 Scaling

Here we have developed an ad-hoc scaling with our dataset. By observing individual traffic patterns, feature values of the dataset were scaled between 0

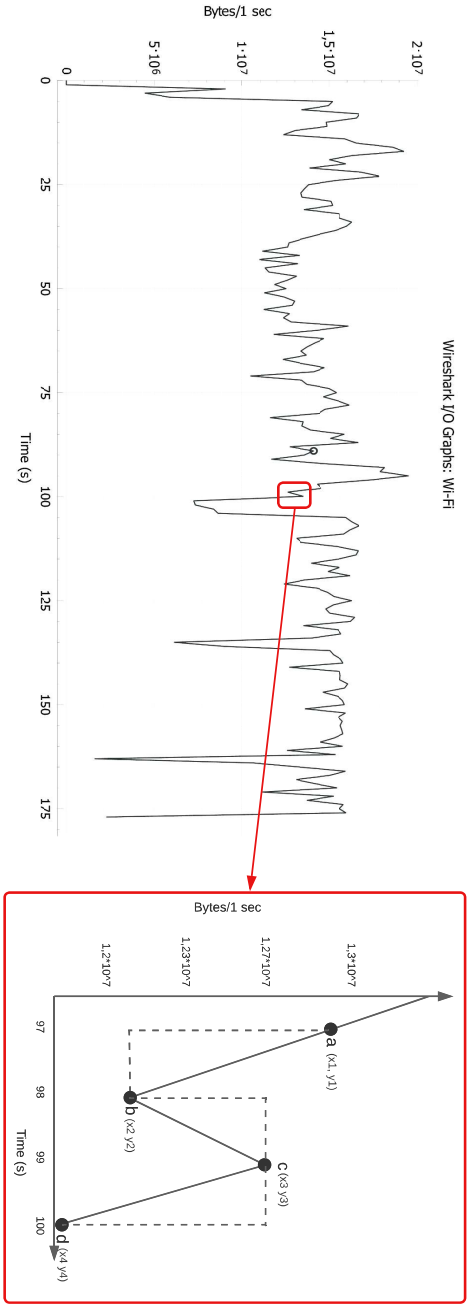

Figure 4.1: Throughput of file transfer traffic and demonstration of trend calculation

to 19. 20 levels have been considered to explain the operation of the proposed method. The scaling system that we have developed is shown in Table 4.4. Here, rank(R) of features DU, DA3, UA3, RDA3, RUA3 are represented by the rank of those features. In the first column, specific ranges for each feature are defined. The ranges are identified by looking at the pattern of each type of traffic. This is a thumb rule for all features, and the table is prepared keeping this in mind.

Number	RDU	RDA3	RUA3	RrDA3	RrUA3
0	<=0,5	<=0,5	<=0,0058	<=0,029	<=0,029
1	0,9	0,75	0,0116	0,058	0,058
.	.	.	.	.	.
18	200	30	0,3	1,5	1,5
19	>=300	>=40	>=0,4	>=3.93	>=3.93

Table 4.4: Scaling the features. RDU is rank of DU

4.2.5 Traffic Types

We took into account three different types of traffic: file transfer, video streaming, and video conferencing. The throughput for the file transfer is generally increasing with time; it is, however, capped by the maximum network capacity. In the case of video streaming, the buffering technique used extracts parts of the available network capacity, which yields very volatile throughput, whereas in video conferencing, applications require a constant rate, typically a few Mbps.

As a second step toward validation, we examined the mixes of several types of traffic to see if low-latency traffic appears in these mixes. We present the

mixtures of traffic types analyzed in Table 4.5.

Traffic Mixes
1 File Transfer, 1 Video Streaming
1 File Transfer, 1 Video Conferencing
1 Video Conferencing, 1 Video Streaming
1 File Transfer, 2 Video Streaming
1 File Transfer, 3 Video Streaming
1 Video Conferencing, 2 Video Streaming
1 File Transfer, 1 Video Conferencing, 1 Video Streaming

Table 4.5: Traffic Mixes

4.2.6 Machine Learning Algorithms

In this work, five machine learning algorithms were used: k-NN, SVM, RF, and C4.5 for traffic classification and k-Means for clustering. We have used k-Means clustering technique to cluster our data into k clusters and thereafter use the resultant cluster label as an additional feature to k-NN and SVM for evaluation of its performance on traffic classification tasks.

4.3 Classification Results and Analysis

In this section, we will show the performance of our classifiers. We will first analyze how different features affect the method's performance and then proceed to present classification accuracy of the method over other methods or baselines.

Finally, detailed model performance is presented, followed by insights into its strengths and weaknesses, including potential areas for improvement.

4.3.1 Selection of Features

The performance of the feature(s) is compared and evaluated using different ML algorithms. We, therefore, used all possible combinations of features at the evaluation step. The results are shown in Table 4.6. Each row gives a different set of features, and the columns have the accuracy scores of the different algorithms over those features.

Feature(s)	Acc (k-NN)	Acc (SVM)	Acc (k-Means)	Acc (C4.5)	Acc (RF)
1 (RDU)	**83,13**	**83,58**	**79,76**	**78,54**	**81,21**
2 (RDA3)	**81,7**	**82,55**	**77,6**	**77,4**	**79,71**
3 (RUA3)	79,46	68,76	74,12	66,13	69,65
4 (RRAD3)	49,31	52,51	48,43	49,23	51,12
5 (RRAU3)	50,09	51,71	50,02	49,62	48,60
1,2	**91,23**	**90,85**	**87,76**	**86,43**	**88,92**
4,5	**52,9**	**53,15**	**50,98**	**50,78**	**51,03**
1,2,3	**94,91**	**93,7**	**90,13**	**89,76**	**90,11**
1,2,3,4	95,71	95,03	93,45	93,78	93,98
1,2,3,4,5	**96,47**	**95,9**	**94,1**	**94,21**	**94,38**

Table 4.6: Comparison of Accuracy (%) for Various Classification Algorithms on Different Feature Combinations

To evaluate the impact of the features for classifying a single traffic type, we used first one feature at a time. We noticed that feature 1 (RDU) is a strong discriminator because it allowed classification by algorithms for individual traffic types with an accuracy of about 83.5%. Similarly, features 2 and 3 have proved to be prominent features. On the other hand, features 4 and 5, which capture the slope between intervals, act as more supportive trend features. Their power can be seen when classifying traffic in a mix.

When features 1, 2, and 3 are used together for classifying single traffic types, k-NN and SVM provided the accuracy of 94.91% and 93.7%, respectively. In most of the cases, k-NN produced the best result among all machine learning algorithms. This could be attributed to the fact that the k-NN model performed well in classifying data into classes based on similarity to neighbors. However, it was also found that feature 1 was the one most needed by the k-NN algorithm, and when this feature was not present, the accuracy scores decreased quite significantly. Finally, the classifiers were trained using features 4 and 5 combined. Of the 4 algorithms, k-NN algorithm registered the highest accuracy of 95.71%, while the k-means could only register a mere 93.45%. The highest obtained accuracy was by the kNN method when trained with all five features. Accuracy of 96.47% was acquired with k-NN and the lowest was from k-means at 94.1% when trained with 5 features used.

4.3.2 Tuning the Machine Learning Algorithms

For the k-NN machine learning algorithm, we began the optimization with the 10-fold cross-validation method to decide on the best sample for the training set.After evaluating the performance of the algorithm across 10 different train-

ing sets, the best set was determined to be the one where $k = 5$ by default. We then proceeded with a hyperparameter tuning procedure on this selection. Finally, we took the parameter $k = 7$ as our selected value. We obtained high classification accuracy for the test dataset after such value selection. For SVM, k-means, C4.5, and RF, the same tuning steps were performed.

4.3.3 Classification Reports for Individual Traffic Types

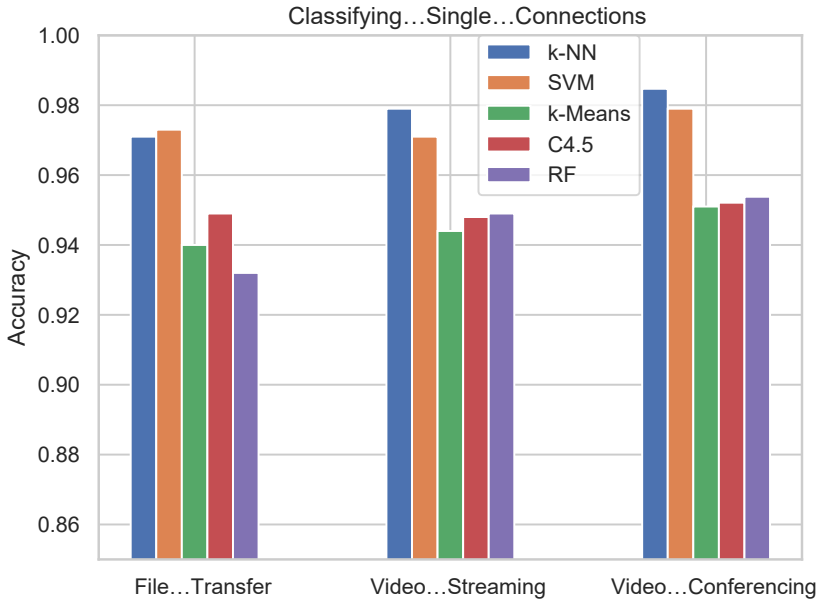

Figure 4.2: Accuracy results of classifying single traffic types.

The performance in classifying the three individual traffic types, of the different algorithms trained with 5 features is demonstrated in Figure 4.2. When the results are analyzed, though all the algorithms provided over 94% accuracy in classifying the file transfer, video streaming, or video conferencing traffic, for the most part, k-NN does 2% better than k-means, C4.5, and RF algorithms. File

transfer, however, is classified quite well by SVM and better than kNN in terms of accuracy, with a very small margin. Since this is the only other place where SVM outperforms kNN, the remaining tasks are performed using only kNN.

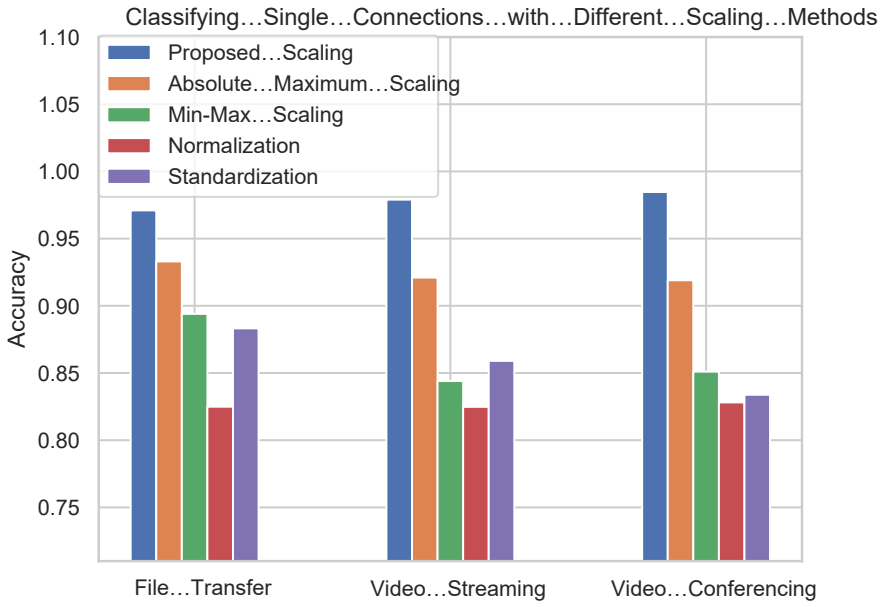

Figure 4.3: Classifying single traffic types with different scaling methods.

Since we decided to proceed with the kNN algorithm, we looked at one more performance metric for our approach. In this next experiment, some of the famous feature scaling techniques were compared with our proposed technique. We evaluated our proposed technique with five independent scaling techniques against the proposed method. From Figure 4.3, it is observed that the proposed method outperforms other scaling methods by above 2% for abs max and by 14% for the normalization method. This is mainly because we manually analyzed patterns for individual traffics in the course of developing the novel scaling system.

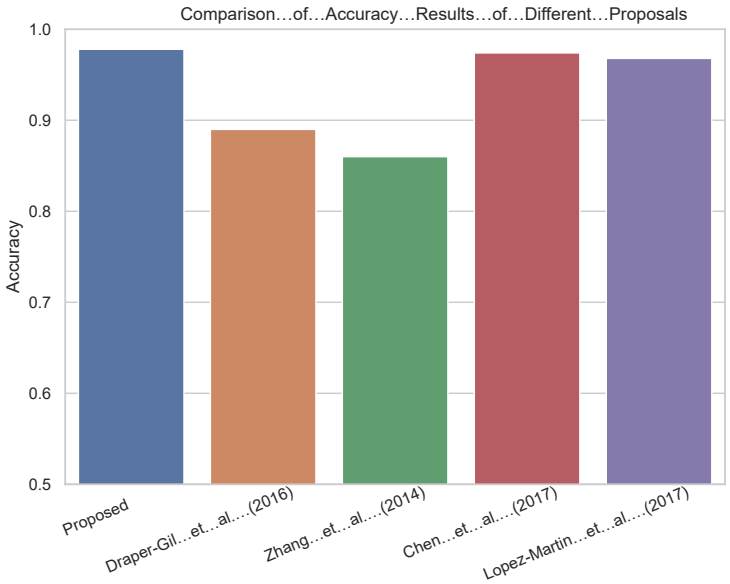

Figure 4.4: Classification accuracy comparison of different approaches for single traffics

Figure 4.4 shows the comparison of our method accuracy results to several published methods. We provide an expanded explanation of these proposals in the related work section. While each proposal has its strengths and weaknesses, our method performed slightly better than those shown in Figure 4.4. In paper Draper-Gil et al. (2016), the author achieved 89% accuracy for the classification of encrypted traffic. On the other hand, in the papers Chen et al. (2017) and Lopez-Martin et al. (2017) authors deployed deep learning tools to classify traffic with accuracy of more than 97%.

In the second experiment, we classified individual types of traffic: we managed to identify the types of traffic and their exact mixes. Our results demonstrate the potential to identify exact traffic mixes, despite the complexity of the task. Figure 4.5 shows that we reach an accuracy of around 84% in identifying

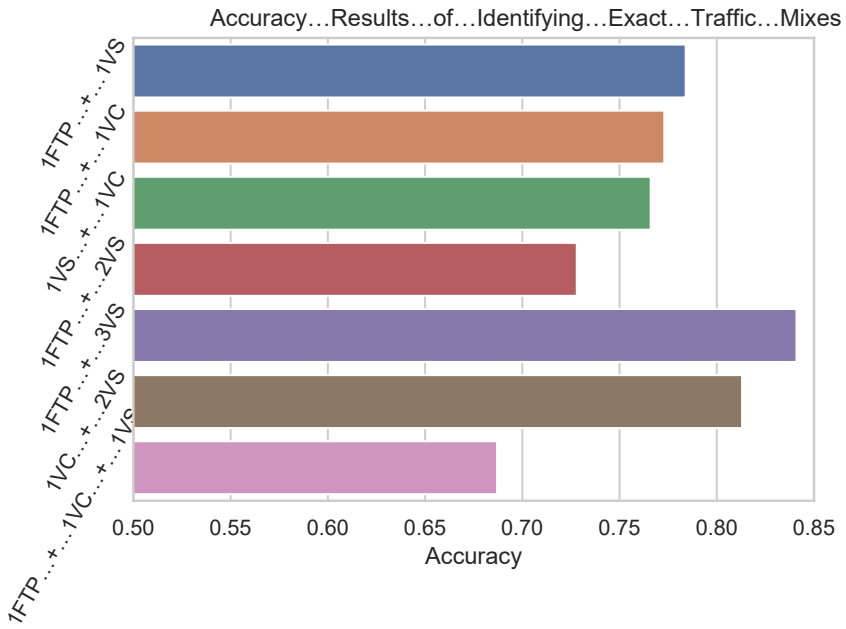

Figure 4.5: Accuracy results of identifying exact traffic mixes

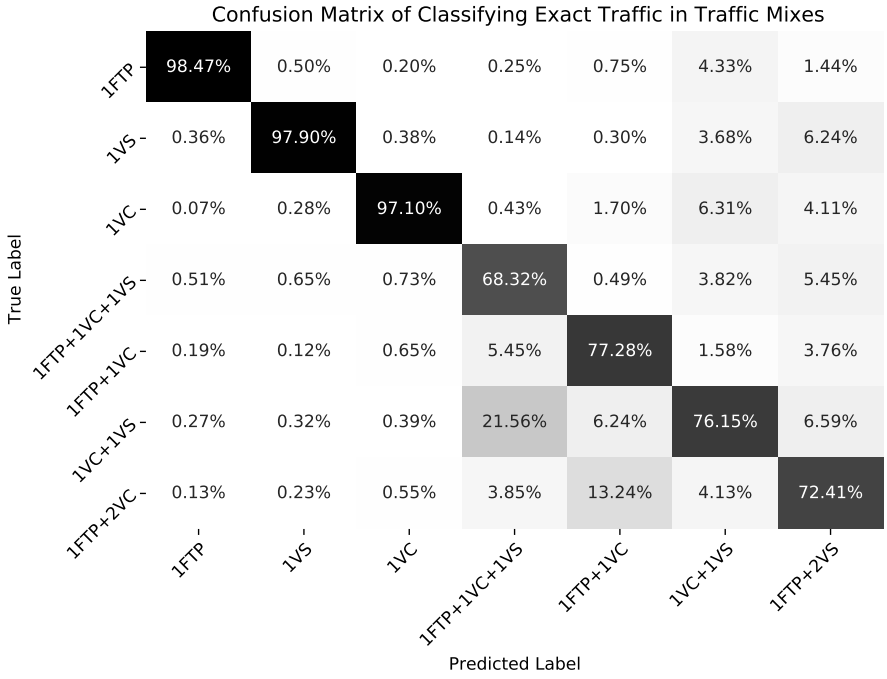

Figure 4.6: Confusion matrix of classifying exact traffic mixes.

mixes of 1FTP+3VS. Furthermore, even by making many traffic flows together, our approach made an identification to the traffic mixes with a high level of accuracy. Results from this experiment are further supported by the confusion matrix shown in Figure 4.6. The results highlight promising performance of our method to classify complex traffic mixes, with individual traffic types being classified with over 97% accuracy. This experiment epitomizes the promise of future research in this field and the great outcomes that can be achieved with persistence.

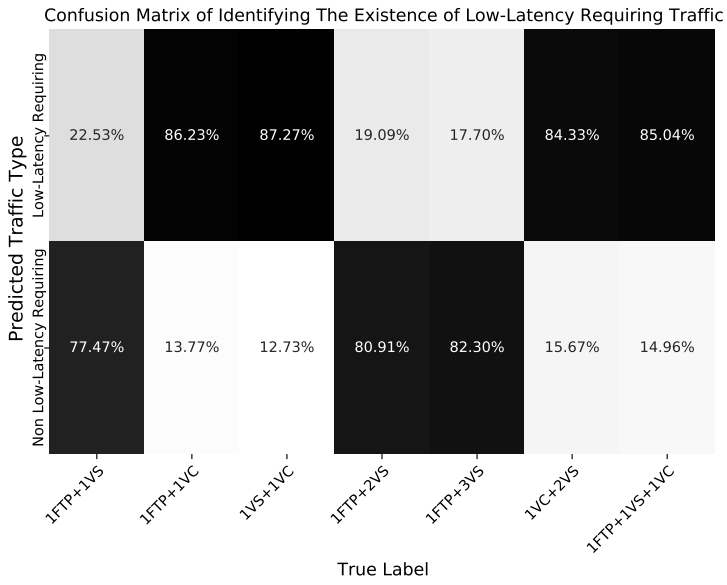

Figure 4.7: Confusion matrix of identifying the existence of low-latency traffic.

In the last experiment, instead of finding the exact nature of a traffic mix, we found whether any low-latency application traffic (video conferencing) is in the traffic mix or not. Video conferencing applications need a constant connection with low latency to maintain the QoE. Results of the experiments are shown in Figure 4.8. The classification results of this experiment are, therefore, much

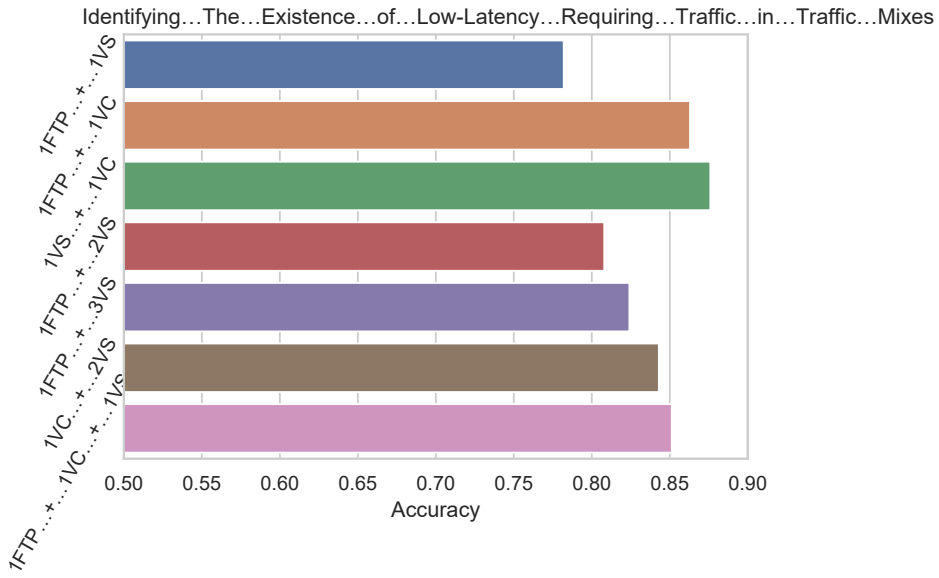

Figure 4.8: Accuracy result of identifying the existence of low-latency requiring traffic in the traffic mix.

better in general. We could achieve accuracies in the range from 78% to 88% while identifying low-latency application traffic within the traffic mix. In Figure 4.7, we finally present the confusion matrix of this classification. Our approach was able to predict the mixed 1VS+1VC as low-latency-requiring traffic with an accuracy rate of 87.27%. Despite the intricate patterns present in file transfer traffic, our method still showed favorable prediction outcomes.

4.4 Conclusion

In this chapter, we introduced a novel approach to low-latency Internet traffic classification by integrating trend-based features into machine learning algorithms. The approach was evaluated on dataset, representing mixed traffic

types such as file transfer, video streaming, and video conferencing, with over 400,000 throughput samples.

The key contributions of this work include the development of five new trend-based features, which capture important traffic patterns over time. These features were integrated into machine learning algorithms, such as k-NN, SVM, Random Forest, and C4.5, and their classification performance was evaluated through a series of experiments. The results demonstrated that the combination of trend-based features significantly improves classification accuracy, particularly for low-latency traffic.

The k-NN algorithm consistently outperformed other models, achieving a maximum accuracy of 96.47% when trained with all five features. Our proposed feature scaling method further enhanced the performance of the classifiers, outperforming traditional scaling techniques by 2-14%. The results of the experiments also showed that our approach achieved over 97% accuracy in classifying individual traffic types within mixed environments, and 84% accuracy in identifying exact traffic mixes. In addition, we demonstrated that our method is highly effective in detecting the presence of low-latency traffic within complex traffic mixes, achieving up to 88% accuracy in identifying low-latency-requiring traffic such as video conferencing. The confusion matrices presented support the robustness of our method in handling intricate traffic patterns, even when multiple traffic types are involved.

This chapter highlights the strength of the proposed methodology in handling both single and mixed traffic types, while maintaining high accuracy across different machine learning algorithms. These results underscore the potential of novel trend-based features and optimized machine learning models for advanc-

ing real-time network traffic classification.

4.5 Future Work

While this chapter has demonstrated the effectiveness of using trend-based features and machine learning models to classify low-latency internet traffic, several avenues remain for future exploration.

- **Integration of Wavelet Transforms:** As outlined in Chapter 5, the inclusion of wavelet-based feature extraction, combined with trend-based features, has shown significant promise in improving classification accuracy. Future work could focus on integrating these two techniques for enhanced performance across a wider range of traffic types. Applying continuous wavelet transforms (CWT) alongside artificial neural networks (ANNs) could further optimize real-time classification.

- **Advanced Deep Learning Models:** In future research, employing more sophisticated deep learning architectures like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and especially Long Short-Term Memory (LSTM) networks could be explored. These architectures are well-suited for handling the sequential and temporal nature of network traffic, which could improve real-time classification accuracy, especially for mixed or encrypted traffic.

- **Real-Time Traffic Analysis:** While the current methodology focuses on offline analysis, moving towards real-time implementation is critical. The next step could involve optimizing computational performance to allow

for deployment in real-time network management systems. Utilizing hardware accelerators such as GPUs or FPGAs may help achieve the required processing speeds without sacrificing classification accuracy.

CHAPTER 5

A Novel Wavelet Transform and Deep Learning-Based Approach for Low-Latency Internet Traffic Classification

In this chapter, we propose a new scheme for classifying low-latency Internet traffic, which combines Wavelet Transform and artificial neural networks. The proposed method is based on the premise that the temporal dynamic properties of low-latency traffic can be viewed as structured "noise" from the perspective of signal processing. By applying Wavelet Transforms to these temporal patterns, we can extract distinctive features that can be utilized to train ANNs for accurate traffic classification. These not only improve the accuracy in the identification of traffic, but also highlight the importance of advanced signal processing in modern network management.

5.1 Motivation

The demand for high-speed Internet services in the digital era has driven the search for effective ways of traffic management and optimization, especially for real-time communication applications such as video conferencing, online gaming, and financial trading systems. While this has been explored in prior chapters, the novelty of this chapter lies in the application of Wavelet Transform-based ANNs for the classification of low-latency Internet traffic. This AI-based approach sets up not only a solid platform for near real-time classification but also shows the importance of modern network management by making use of such advanced methodologies.

The key contribution of this chapter is the use of Wavelet Transforms to capture and analyze the temporal dynamic properties of low-latency traffic. These traffic patterns can be compared to structured "noise" in signal processing, where Wavelet Transforms allow for the decomposition of the signal into its fundamental time-frequency domain elements. This enables the extraction of distinctive features that are then used to train ANNs for more precise classification. This method significantly enhances the ability to identify and classify low-latency traffic in complex environments, where traditional approaches may fall short. By integrating Wavelet Transforms and ANNs, this chapter provides a new perspective on managing low-latency traffic, emphasizing the importance of advanced signal processing in modern network management. The proposed methodology offers a robust platform for near real-time classification, paving the way for more adaptive and responsive network systems.

In the following sections, we detail our approach as represented in Figure

5.1, discuss the process of collecting data, present our experimental results, and provide thoughtful discussions. The goal is to illustrate the efficacy of our approach in classifying and identifying low-latency traffic.

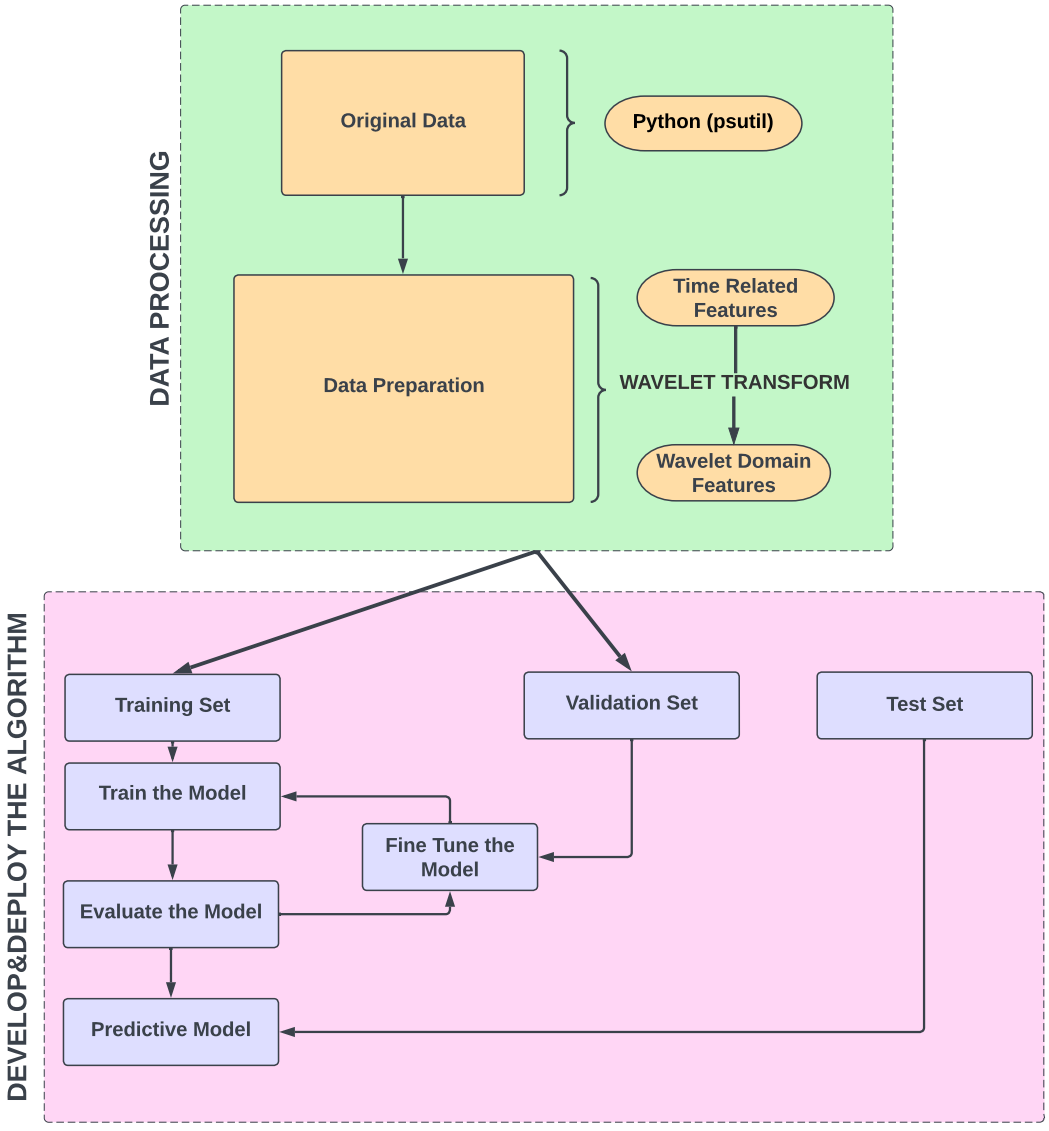

Figure 5.1: System Architecture.

5.2 Methodology

In this section, we detailed the methodology for the identification of low-latency Internet traffic. Our approach combined Artificial Neural Networks with the application of Wavelet Transform to create a powerful system for distinguishing low-latency traffic from other types of network traffic.

5.2.1 Data Collection

The dataset used in this chapter is an expanded version of the dataset from Chapter 4. While the dataset in Chapter 4 was sufficient for trend-based feature extraction and machine learning models, it was not complex enough to fully capture the temporal dynamics required for wavelet transform and deep learning analysis. To address this, we expanded the dataset by collecting additional traffic data, including more diverse and granular time-series measurements.

The expanded dataset includes three categories of traffic:

- **Low-latency traffic** (e.g., video conferencing, online gaming),
- **Non-low-latency traffic** (e.g., file transfer, video streaming), and
- **Mixed traffic** (a combination of low-latency and non-low-latency traffic).

The data collection process involved recording throughput in both downlink and uplink directions in real-time, resulting in over 350,000 samples of throughput values (more than 35 hours of application usage). This expanded dataset provides a more comprehensive representation of network traffic, enabling a more accurate analysis of low-latency traffic patterns using wavelet transforms and deep learning models.

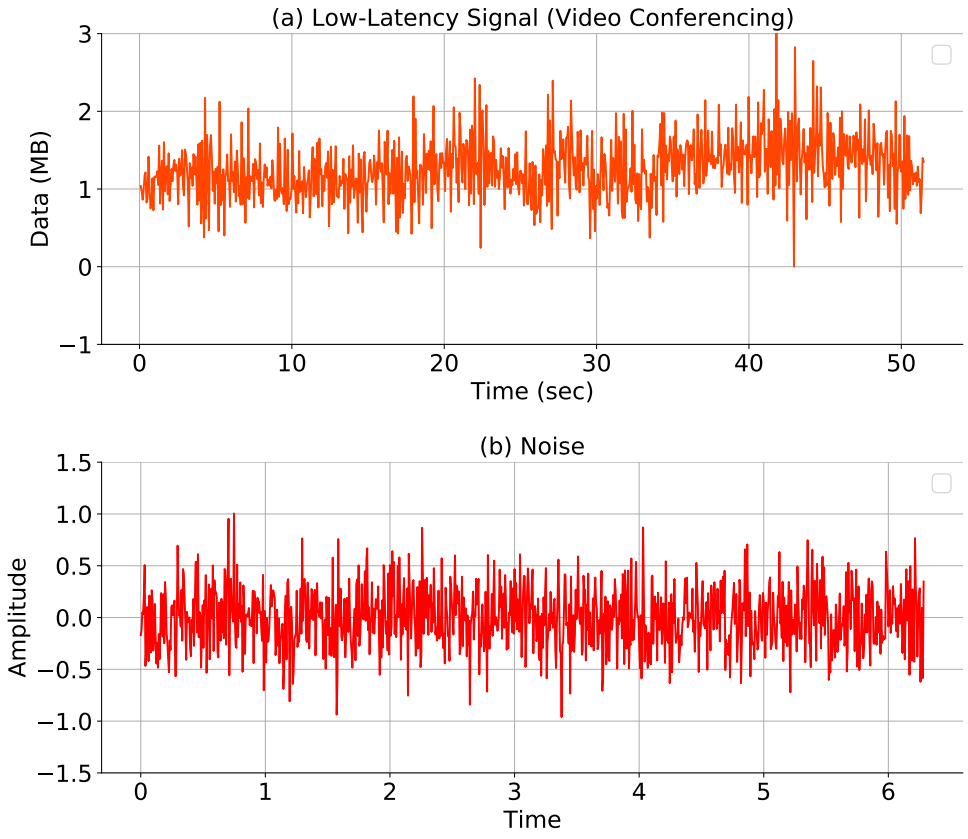

Figure 5.2: Parallel Patterns: (**a**) Low-latency traffic and (**b**) gaussian noise.

5.2.2 Introducing Continuous Wavelet Transform (CWT) with Ricker Wavelet

As it was shown in previous sections, from a statistical point of view, low-latency traffic demonstrates some relevant features similar to those of Gaussian noise, which is a strong motivation for Wavelet Transform techniques. In this regard, it is relevant to discuss the complexity of network data flows. In Figure 5.2, (a) Low-latency Internet traffic patterns of video conferencing (zoom call) and (b) Gaussian noise (python generated) are demonstrated. The similaritiy of these patterns leads to interesting insights about low-latency traffic that may exhibit

noise-like characteristics. In signal processing, noise is conventionally viewed as unpredictable fluctuations or disturbances masking meaningful information in a signal Zou et al. (2016). In contrast, the low-latency traffic has a kind of ordered 'noise,' some kind of confined and organized flow of data with frequent exchanges of packets.

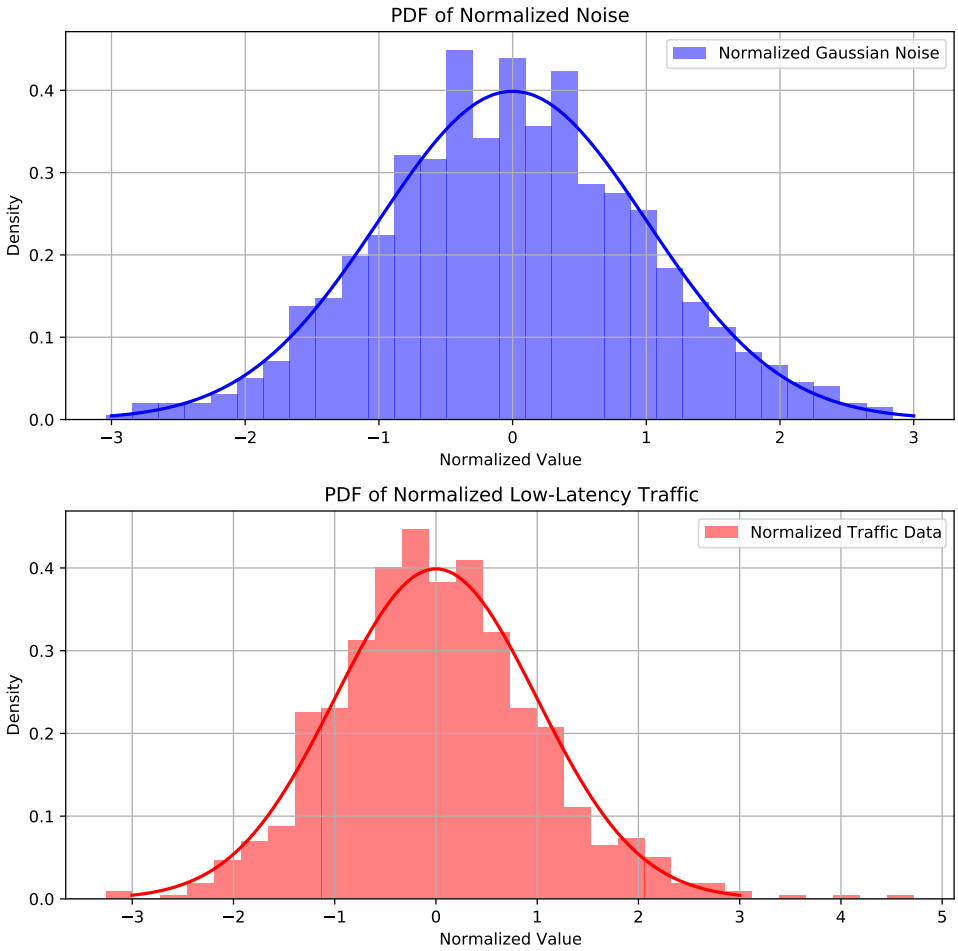

Figure 5.3: Parallel Distributions: (**a**) Gaussian noise and (**b**) low-latency traffic.

In addition, to support the observation made, Figure 5.3 shows the Probability Distribution Function (PDF) of both low latency traffic (b) and Gaussian

noise (a). The analysis of PDF proves that both signals have similar properties: classic bell-shaped curve, maximum density, and tail of the distribution when normalized. These similarities, in turn, strongly reinforce the idea that traditional signal processing tools such as the Wavelet Transform could be used for low-latency traffic classification. This parallel also points to the fact that while the data exchange in the low-latency traffic might look like some sort of noise for a casual observer, it is actually a meaningful and structured element within the network environment, thus making Wavelet Transforms a suitable tool for its analysis and classification Kontogeorgaki et al. (2017).

Using Gaussian noise as an analogy for low-latency traffic offers several advantages:

- **Statistical Framework for Analysis:** Gaussian noise provides a statistical framework to analyze similarities in temporal dynamics. This perspective allows for the identification of characteristic features within traffic patterns that might otherwise be obscured in raw data.

- **Applicability of Signal Processing Tools:** While low-latency traffic exhibits noise-like characteristics, it is the structured nature of the traffic that makes wavelet transforms effective. This structured behavior enables advanced signal processing techniques, such as wavelet transforms, to extract meaningful patterns by focusing on time-frequency localized features.

- **Enhanced Feature Extraction:** The analogy highlights temporal and statistical properties of traffic patterns, such as variability, density, and frequency distributions, which are unique to network traffic. These features do not strictly adhere to Gaussian noise characteristics but rather provide

a framework for capturing complex behaviors in traffic data. This distinction is pivotal for training machine learning models to classify traffic accurately.

- **Improved Classification Accuracy:** By leveraging the structured noise analogy, classifiers can distinguish low-latency traffic from other types more effectively. This approach is particularly beneficial in mixed and encrypted environments where traditional classification methods often struggle.

- **Insight into Temporal Dynamics:** The analogy highlights the temporal dynamics of low-latency traffic, such as rapid packet exchanges and short-term variability. Understanding these dynamics is crucial for real-time traffic management and optimization.

Figures 5.2 and 5.3 illustrate the similarities between low-latency traffic and Gaussian noise, showcasing their parallel patterns and probability density functions. These comparisons reinforce the validity of the analogy and demonstrate its utility in developing advanced traffic classification methodologies.

The Gaussian noise analogy, when combined with tools like wavelet transforms and artificial neural networks, forms the foundation for the proposed methodology in this chapter. By leveraging this perspective, the thesis emphasizes the importance of advanced signal processing in modern network management, paving the way for more accurate and adaptive solutions to real-time traffic classification challenges.

Obtaining representations of the features in the frequency domain is very important for several reasons. The frequency domain gives a different view of

the data. It emphasizes those patterns and features that might not be easily observable in the time domain. It provides the basis upon which analysis of non-stationary signals is preferred, since properties under study continuously change with time. All of the extracted time-related features undergo processing with Continuous Wavelet Transform. Figure 5.4 highlights these time-related features and their transformation to wavelet domain.

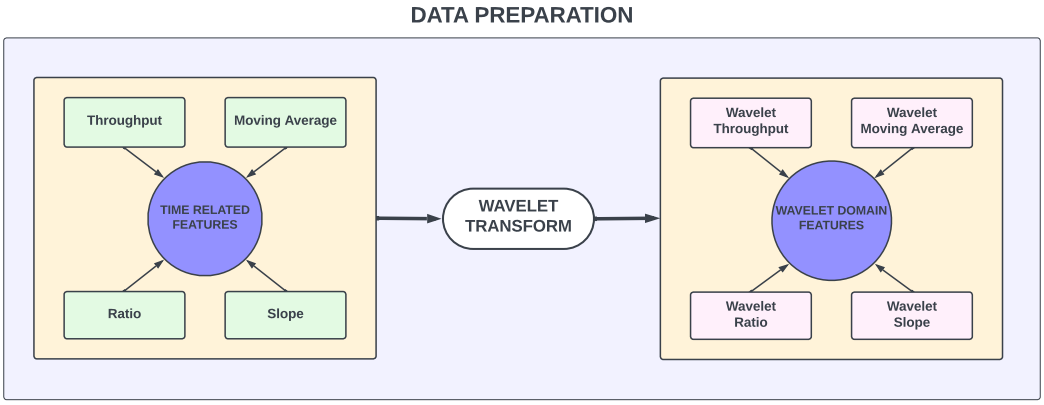

Figure 5.4: Features from time domain to wavelet domain

The CWT is a very strong mathematical tool to analyze the non-stationary signal in time and frequency domains. It differs from the Fourier Transform in that it decomposes the same signal into various frequency components at real-time, thus enabling the extraction of time-frequency localized features and time-varying characteristics. Apparently, that captures fine-scale variations and changes of the signal—the reason why CWT is particularly well-suited for analyzing dynamic and irregular signals. The time characteristics of such signals are alike with those of low-latency Internet traffic; thus, CWT acts as a good means for its classification.

The Ricker wavelet (also called the Mexican hat wavelet) is one of the com-

monly used wavelets in CWT. The Ricker wavelet is defined mathematically as follows:

$$\text{Ricker}(t, \sigma) = (1 - 2\pi^2 f^2 t^2) e^{-\pi^2 f^2 t^2}, \tag{5.1}$$

where t is the time variable, f is the central frequency of the wavelet, and σ is the width parameter that determines the scale of the wavelet.

By applying a Wavelet Transform, the feature representation enhances in such a way that time-related features are decomposed into distinct frequency components. In fact, the transformation enables the model to extract pattern information from various frequency bands, hence it is possible to capture both short-term and long-term variations in network behavior. This aspect is crucial for proper and reliable classification of low-latency Internet traffic because, by the nature of real-time applications, the patterns are usually complex and multiscale.

Furthermore, low-latency traffic such as video conferencing or online gaming, often involves fluctuations in throughput that may resemble noise when analyzed using conventional time-domain methods. However, by applying Wavelet Transform, we can isolate and emphasize these fluctuations in throughput, which are characteristic of low-latency traffic, thus distinguishing them from other traffic types such as file transfer or video streaming, which have different temporal dynamics. For instance, the Wavelet Transform helps to separate high-frequency components corresponding to the rapid exchanges in low-latency traffic from the slower, more consistent patterns seen in bulk data transfers. Figures 5.5 and 5.6 illustrate this transformation for throughput and the ratio of sent to received packets, respectively, highlighting the distinct frequency components that the

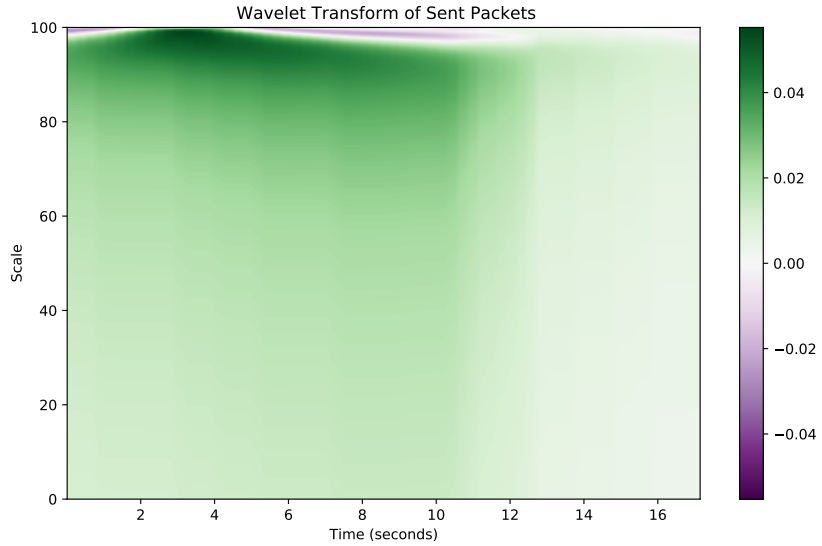

Figure 5.5: Throughput in wavelet domain.

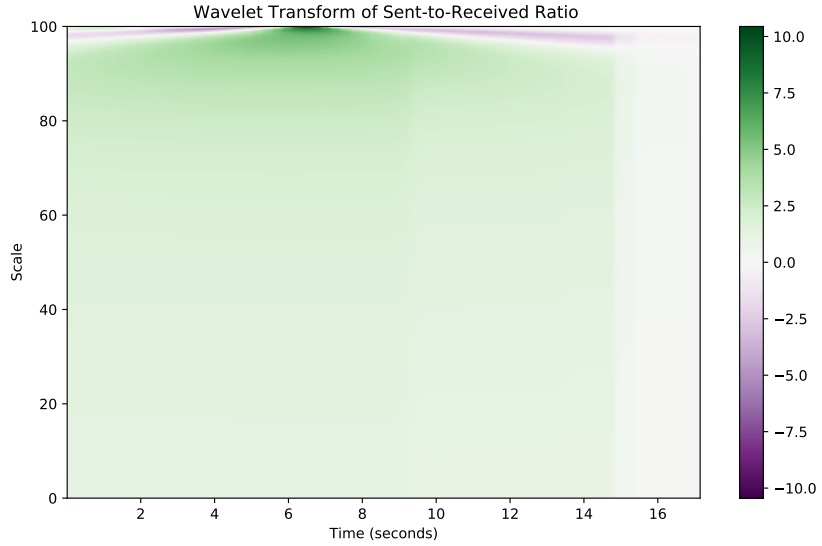

Figure 5.6: Ratio in wavelet domain.

Wavelet Transform extracts from the raw time-domain features.

Using these enhanced feature representations, our ANN model becomes more adept at recognizing the unique patterns associated with low-latency traffic. This addresses the core challenge in traffic classification, where the goal is to accurately identify and prioritize time sensitive data flows among a diverse mix of network activities.

5.2.3 Data Preparation

After collecting raw data related to sent and received packets, a series of pre-processing steps were applied to extract novel time-domain features. These pre-processing steps were designed to improve the data set with meaningful features that could be used for effective classification. The following time-domain features were derived for each network traffic, and their relevance to the research objectives is highlighted:

Throughput

This feature represents the total volume of data (in bytes) captured within specific time intervals during network traffic. The collected data includes the size (in bytes) of individual sent and received packets aggregated over these intervals. The features are denoted by S_i for the total volume of the sent packets and R_i for the total volume of received packets within the interval. Here, i represents the index of the time interval. Thus, for each traffic flow, the feature 'Throughput' captures the volume of sent and received packets, providing valuable information about the data traffic patterns over time.

Relevance to Research: Throughput patterns help identify the overall data

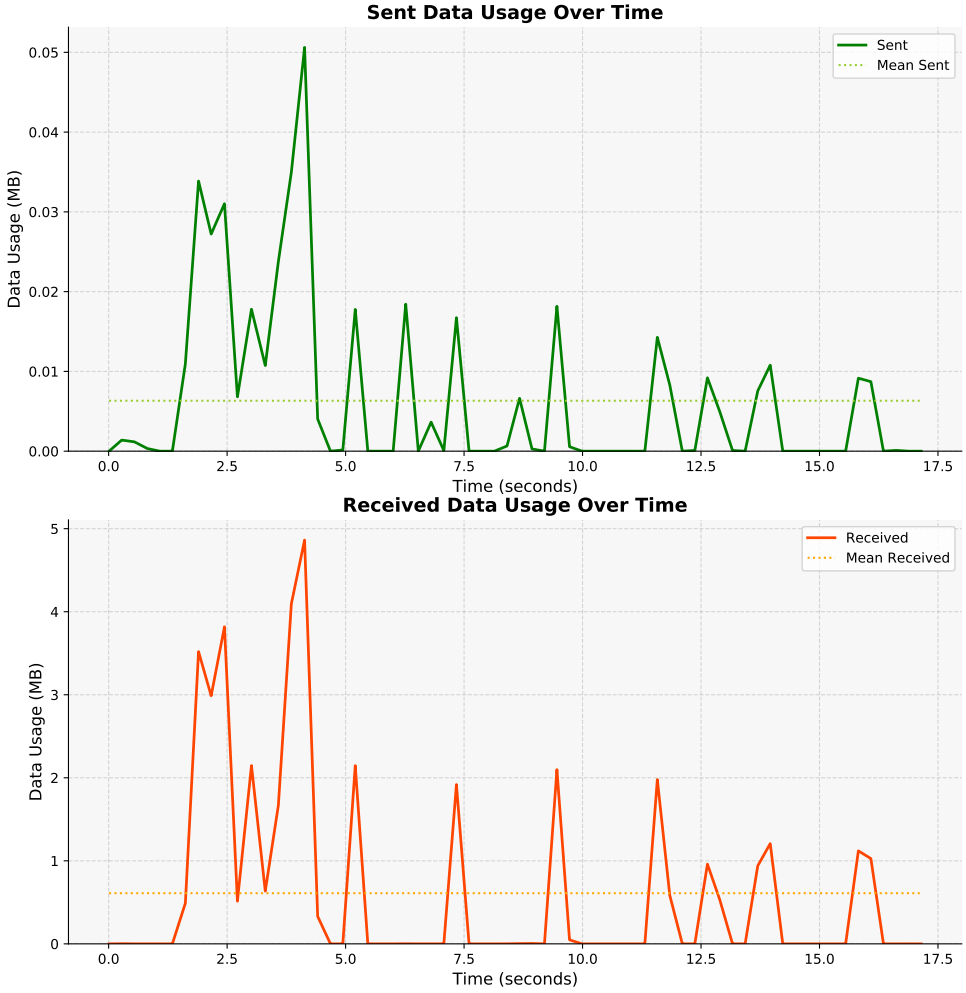

Figure 5.7: Throughput of YouTube traffic.

flow dynamics, distinguishing high-throughput activities such as file transfers from steady and lower-throughput activities like video conferencing. This supports the classification of low-latency traffic amidst mixed environments.

In Figure 5.7, the throughput of YouTube traffic in the downlink and uplink is demonstrated.

Moving Averages

This feature calculates the moving average of throughput in time, which can provide insight into the application's traffic patterns. The moving averages of throughput in uplink and downlink are denoted by MA_{Sent}, $MA_{Received}$, respectively, and computed using the following formulas:

$$MA_{sent}[i] = \frac{1}{k} \sum_{j=i-k+1}^{i} N_{\text{sent}}[j], \tag{5.2}$$

$$MA_{\text{received}}[i] = \frac{1}{k} \sum_{j=i-k+1}^{i} N_{\text{received}}[j], \tag{5.3}$$

where k represents the moving average period (window size), i denotes the current time index, and $N_{\text{sent}}[j]$ represents the throughput at that time j.

Relevance to Research: Moving averages smooth short-term fluctuations, uncovering long-term trends. This is particularly useful in identifying steady traffic patterns in low-latency applications, helping the model focus on consistent behaviors critical for accurate classification.

Figure 5.8 shows the moving average of YouTube traffic.

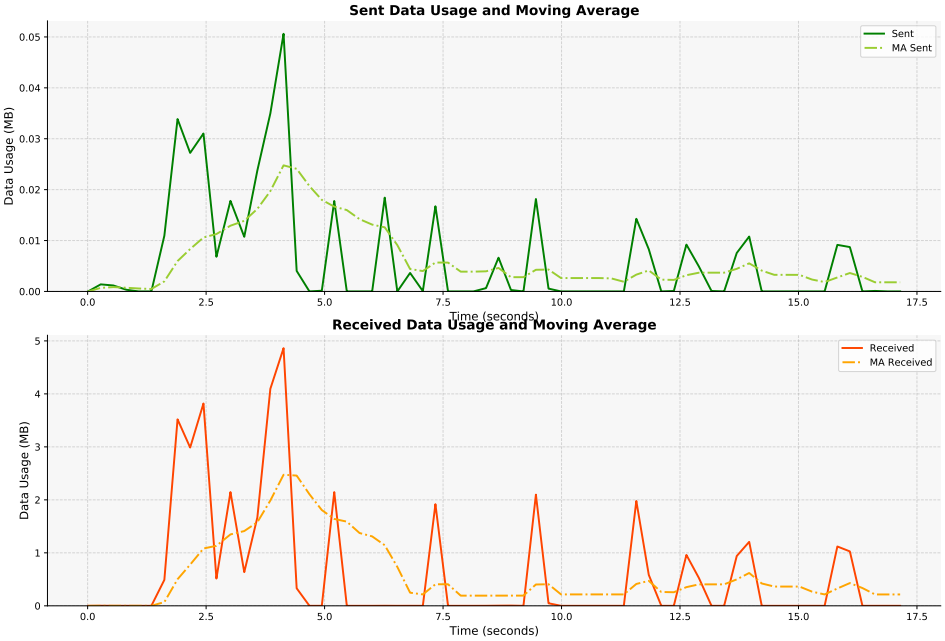

Figure 5.8: Moving average of YouTube traffic.

Ratio

This feature measures the downlink to uplink throughput values over time. The ratio, denoted by R, is calculated as:

$$R[i] = \frac{N_{received}[i]}{N_{sent}[i]}, \tag{5.4}$$

where i denotes the current time index, $N_{\text{received}}[i]$ represents the downlink throughput, and $N_{\text{sent}}[i]$ represents the uplink throughput at time i.

Relevance to Research: This ratio highlights the balance or asymmetry in traffic. Low-latency traffic often involves frequent bidirectional exchanges, while other types, such as video streaming, tend to have unidirectional data flow. This feature helps differentiate such behaviors effectively.

Figure 5.9: Received to sent throughput ratio of YouTube traffic.

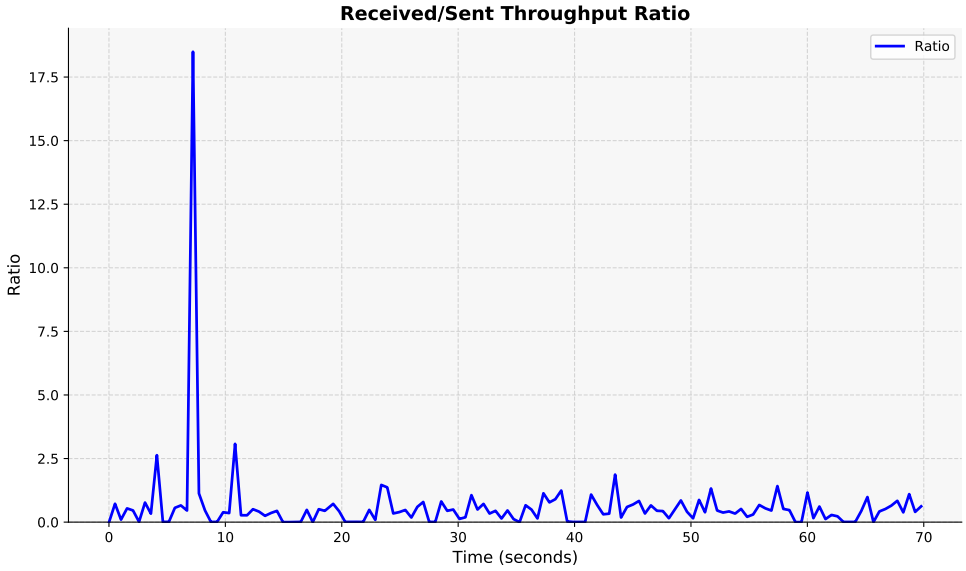

Figure 5.9: Downlink/Uplink throughput ratio.

Slope

The slope of the throughput indicates the change in the number of packets over time. It is represented by S_{sent} for sent packets, $S_{received}$ for received packets and calculated as:

$$S_{sent}[i] = \frac{N_{sent}[i] - N_{sent}[i-1]}{t[i] - t[i-1]}, \tag{5.5}$$

$$S_{received}[i] = \frac{N_{received}[i] - N_{received}[i-1]}{t[i] - t[i-1]}, \tag{5.6}$$

where $N_{sent}[i]$ represents the number of sent packets at time i, and $t[i]$ is the corresponding time stamp.

Relevance to Research: The slope captures the rate of change in traffic flow, highlighting transient trends. This feature is critical for detecting dynamic behaviors in low-latency traffic, such as bursts during interactive sessions or

steady declines during buffering.

The slope feature provides crucial discriminatory information toward classification of various classes of low-latency Internet traffic. A representation can be found in Figure 5.10.

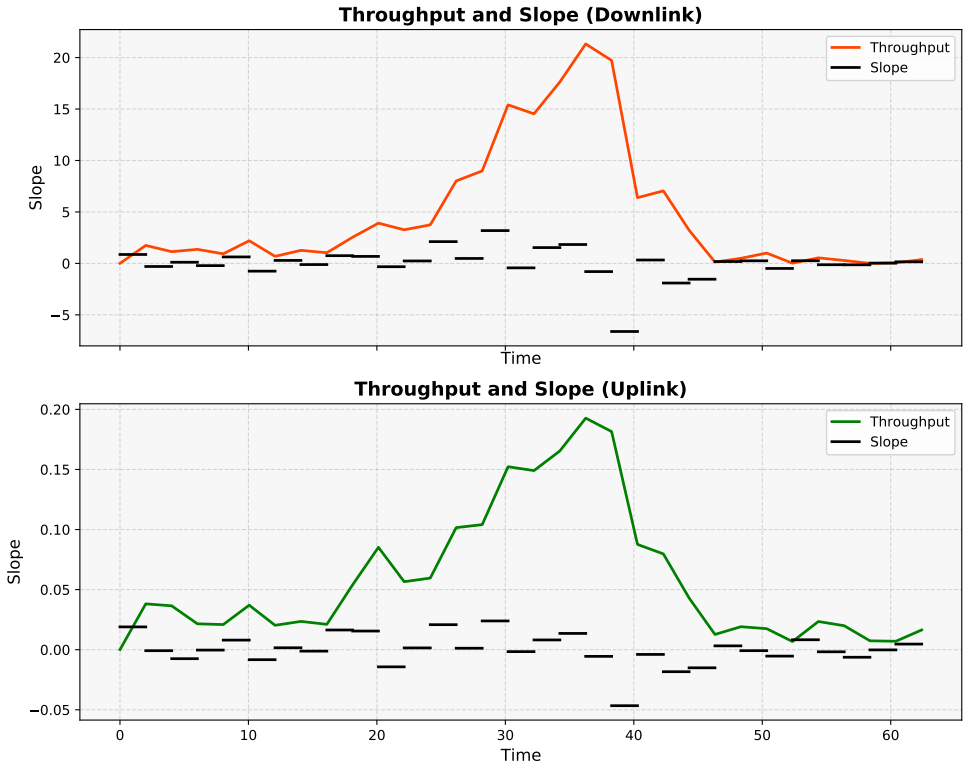

Figure 5.10: Slope of throughput for downlink and uplink traffic.

To provide a clear overview, Table 5.1 summarizes the proposed time-domain features, their purpose, and their relevance to the research objectives.

5.2.4 Artificial Neural Network

The rationale for the employment of an MLP in this framework is to take advantage of its powerful pattern recognition capability Hammedi (2021) for accurate

Feature	Purpose	Relevance to Research
Throughput	Measures the total data flow in bytes over time in- tervals.	Differentiates high-throughput activities (e.g., file transfers) from steady, low-throughput applications (e.g., video confer- encing).
Moving Average	Smooths short-term fluctuations to reveal long-term trends.	Identifies steady traffic behaviors critical for low-latency applications.
Ratio	Measures the balance between downlink and uplink traffic.	Highlights bidirectional traffic patterns typical of low-latency applications, distinguishing them from unidirectional traffic types.
Slope	Captures the rate of change in traffic flow over time.	Detects dynamic traffic behaviors, such as bursts or declines, essential for classifying low-latency traffic

Table 5.1: Summary of Time-Domain Features and Their Relevance

classification in low-latency Internet traffic. The MLP serves as the core part of the artificial neural network model, which provides the backbone for conducting temporal dynamics analysis on complex-patterned network traffic data. In this respect, MLP shall be used to efficiently classify the Internet traffic types and achieve high accuracy in identifying the low-latency traffic patterns. The multilayer perceptron is included in a class of feed-forward artificial neural networks as illustrated in Figure 5.11. Normally, it consists of three or more layers. The first layer is used to input data. Further, one or more hidden layers are allowed for feature abstraction from the input data. The final layer generates a classification output. Each hidden layer, such as i^{th} layer, consists of many neurons mainly using a nonlinear activation function being with the following expression form:

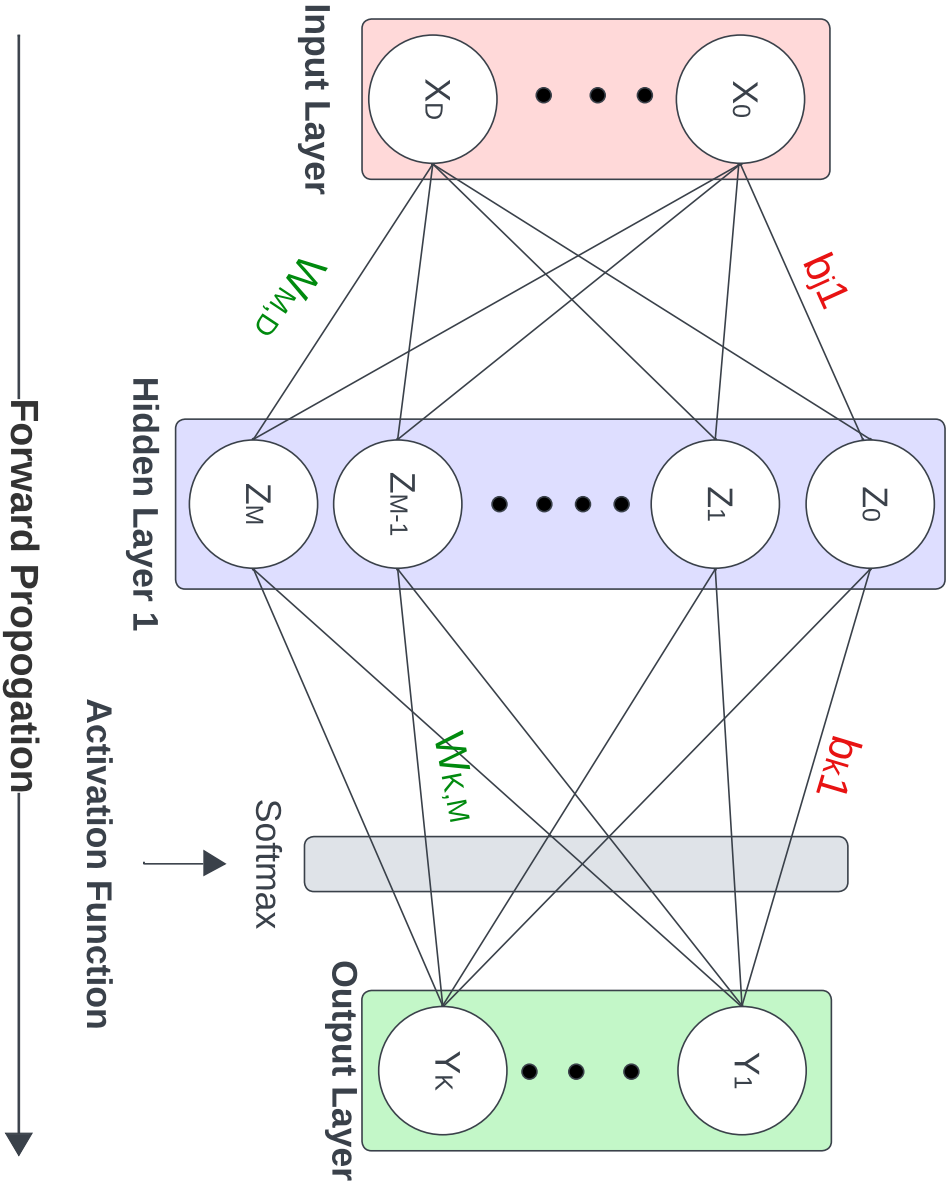

Figure 5.11: Overview of the MLP based ANN classifier.

$$f(x) = \sigma(W^{(i)} \cdot x + b^{(i)}), \tag{5.7}$$

where $\sigma(\cdot)$ denotes an activation function, for example, $\sigma(x) = max(0, x)$. A key attribute of the activation function is its ability to provide a smooth transition as input values change. $W^{(i)}$ represents a weight matrix, and $b^{(i)}$ is a bias vector. It is possible to have multiple hidden layers, and each layer performs the same function but with distinct weight matrices and bias vectors. The final layer produces the output based on the results of the last hidden layer, often denoted as layer j and described as:

$$\sigma(x) = g(W^{(j)} \cdot x + b^{(j)}) \tag{5.8}$$

The deep learning model employed in this study is a feedforward neural network with three layers as demonstrated in Figure 5.11: an input layer, two hidden layers, and an output layer. The architecture is mathematically defined as follows.

- **Input Layer:** The input layer consists of N neurons, where N represents the number of features in the dataset. These features include throughput T, moving averages MA, downlink-to-uplink ratio R, slope S and Wavelet Transformed counterparts of them WT. The input vector X is represented as:

  $$X_{input} = [T_i, MA_i, R_i, S_i, WT_i], \tag{5.9}$$

  where $X_i$represents the value of the feature i.

- **First Hidden Layer:** The first hidden layer has H_1 neurons, where $H_1 = 8$ in this implementation. Each neuron applies the Rectified Linear Unit (ReLU) activation function $f_{\text{ReLU}}(x) = \max(0, x)$ to its weighted sum of inputs. Mathematically, for each neuron j in the first hidden layer:

$$Z_j^{(1)} = \sum_{i=1}^{N} W_{ij}^{(1)} \cdot X_i + b_j^{(1)}, \tag{5.10}$$

$$A_j^{(1)} = f_{\text{ReLU}}(Z_j^{(1)}), \tag{5.11}$$

  where W_{ij}^1 represents the weight of the connection between $b_j^{(1)}$ is the bias term for neuron j.

- **Second Hidden Layer:** The second hidden layer consists of H_2 neurons, where $H_2 = 4$ in this configuration. Similar to the first hidden layer, each neuron applies the ReLU activation function to its weighted sum of inputs. Mathematically, for each neuron k in the first hidden layer:

$$Z_k^{(2)} = \sum_{j=1}^{H_1} W_{jk}^{(2)} \cdot A_j^{(1)} + b_k^{(2)}, \tag{5.12}$$

$$A_k^{(2)} = f_{\text{ReLU}}(Z_k^{(2)}), \tag{5.13}$$

  where W_{ij}^1 represents the weight of the connection between $b_j^{(1)}$ is the bias term for neuron j.

- **Output Layer:** The output layer comprises C neurons, where C is the different number of traffic classes in the dataset. It uses the softmax

activation function to produce class probabilities. The softmax function calculates the probability $P(Y_i)$ of each class i, given the input features. Mathematically, for each class i in the output layer:

$$Z_i^{(0)} = \sum_{k=1}^{H_2} W_{ki}^{(0)} \cdot A_k^{(2)} + b_i^{(0)} \tag{5.14}$$

$$P(Y_i) = \frac{e^{Z_i^{(0)}}}{\sum_{j=1}^{C} e^{Z_j^{(0)}}} \tag{5.15}$$

The class with the highest probability is selected as the predicted class. This architecture enables multi-class classification, making it suitable for low-latency network traffic identification.

5.3 Experiment Setup

Owing to the simplicity and efficiency of our design, experiments were performed on a standard Windows 10 PC with an Intel Core i7 running at 3.20 GHz with 16 GB of RAM. In fact, this configuration was enough for executing and training the artificial neural network model without being in need of additional computational resources such as GPU acceleration. Moreover, we used TensorFlow Abadi et al. (n.d.), which is a flexible and powerful machine learning framework, for even greater ease in the implementation and effective training of our model.

5.3.1 Data Set and Traffic Scenarios

The expanded dataset used in this chapter consists of three main traffic types: file transfer, video streaming, and low-latency traffic. Each traffic type was sampled for approximately one hour, resulting in a balanced dataset with 14,679 samples for file transfer, 14,287 samples for video streaming, and 14,510 samples for low-latency traffic. The exact sample sizes are provided in Table 5.2. This balanced representation ensures that the model is trained and tested on an equal number of samples for each traffic type, reducing the risk of bias in the classification results.

Traffic Type	Duration	Total Samples
File Transfer	∼1 Hour	14,679
Video Streaming	∼1 Hour	14,287
Low-Latency	∼1 Hour	14,510

Table 5.2: Traffic Types and Sample Sizes

To evaluate the performance of our model, we designed a series of traffic scenarios that reflect real-world network conditions. These scenarios were inspired by the Low Latency DOCSIS study White et al. (2019), which evaluates latency performance in mixed traffic environments. However, our scenarios were tailored to better capture the unique challenges of low-latency traffic classification using wavelet transforms and deep learning.

The traffic scenarios are summarized in Table 5.3 and Table 5.4. Table 5.3 lists the basic scenarios, which involve pairwise combinations of traffic types, while Table 5.4 describes more complex scenarios involving three or more types

Traffic Scenario	Traffic Types	Low-Latency Traffic Present
A+B	File Transfer + Video Streaming	No
A+C	File Transfer + Low-Latency	Yes
B + C	Video Streaming + Low-Latency	Yes
A + A	File Transfer + File Transfer	No
B + B	Video Streaming + Video Streaming	No
C + C	Low-Latency + Low-Latency	Yes

Table 5.3: Basic Traffic Scenarios

Traffic Scenario	Traffic Types	Low-Latency Traffic Present
3A	3 × File Transfer	No
3B	3 × Video Streaming	No
3C	3 × Low-Latency	Yes
2A+B	2 × File Transfer + Video Streaming	No
A+2B	File Transfer + 2 × Video Streaming	No
2A+C	2 × File Transfer + Low-Latency	Yes
A+2C	File Transfer + 2 × Low-Latency	Yes
2B+C	2 × Video Streaming + Low-Latency	Yes
B+2C	Video Streaming + 2 × Low-Latency	Yes
A+B+C	File Transfer + Video Streaming + Low-Latency	Yes
2A+B+C	2 × File Transfer + Video Streaming + Low-Latency	Yes
A+2B+C	File Transfer + 2 × Video Streaming + Low-Latency	Yes
A+B+2C	File Transfer + Video Streaming + 2 × Low-Latency	Yes

Table 5.4: Complex Traffic Scenarios

of traffic coexisting simultaneously.

These scenarios were chosen to comprehensively test the model's ability to classify low-latency traffic in both simple and complex environments. The inclusion of mixed traffic scenarios ensures that the model is evaluated under conditions that closely resemble real-world network behavior.

5.3.2 Evaluation Metrics

As carried out by Lotfollahi et al. (2020), the performance of the model was gauged using key metrics (accuracy, precision, recall, F1 score) that effectively classified traffic. Accuracy, A, was used to estimate the actual performance, which indicated the ratio between correctly classified samples and the total number of samples. It was calculated using equation 5.16. However, it was important to consider that high accuracy on its own might not provide the full picture, particularly when datasets were imbalanced.

$$A = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \tag{5.16}$$

Precision and recall were useful metrics that gave even more insight into the performance of the model on the different classes of traffic. Here was where precision was important: precision referred to the ratio between true positive predictions and the total number of positives predicted, showing (or which showed) the reliability of the model in identifying low-latency traffic with no false alarms. Whereas recall—which was the ratio of true positive predictions to all actual positives—concerned the contribution the model made to capturing all the relevant low-latency traffic instances. The mathematical representation of such metrics, along with their values, was given by:

$$P = \frac{T_P}{T_P + F_P} \tag{5.17}$$

$$R = \frac{T_P}{T_P + F_N} \tag{5.18}$$

The F1 score, which is the harmonic mean of precision and recall, gives a well-balanced measure that can be very useful in cases where the classes are highly imbalanced. In our study, therefore, the F1 score, being a robust metric of the overall effectiveness for the proper identification of low-latency traffic, is defined as:

$$F_1 = \frac{2PR}{P + R} \tag{5.19}$$

Focusing on these metrics, we get a better sense of our classifier's strengths and weaknesses. For instance, while accuracy gives the big picture, precision and recall allow us to zoom into aspects of model performance that are critical for practical applications requiring minimized false positives in low-latency traffic detection. This ensures our model is accurate, reliable, and efficient for practical scenarios.

5.3.3 Hyperparameter Tuning

In pursuit of optimizing the performance and robustness of our ANN model, a systematic hyperparameter tuning process was conducted. Table 5.5 presents the hyperparameter tuning process for the ANN model used in this study. This includes the number of hidden layers, the activation function, the learning rate, batch size, number of epochs, optimizer, and dropout rate. Below, we provide

a detailed explanation of how each hyperparameter affects the model's performance and the rationale behind the chosen values.

Hyperparameters	Range	Selection
Number of hidden layers	[1, 2, 3, 4, 5]	2
Activation Function	[sigmoid, tanh, ReLU]	ReLU
Learning Rate	[0.1, 0.01, 0.001]	0.001
Batch Size	[16, 32, 64, 128]	32
Number of Epochs	[10, ..., 50, ..., 150]	100
Optimizer	[Adam]	Adam
Dropout Rate	[0, 0.1, 0.2, 0.3, 0.4, 0.5]	0

Table 5.5: Hyperparameter tuning for ANN model

Number of Hidden Layers: The number of hidden layers in a neural network determines the depth of the model, which in turn affects its ability to learn complex patterns in the data. A model with too few layers may underfit, failing to capture the underlying patterns, while a model with too many layers may overfit, capturing noise in the training data. In our experiments, we tested models with 1 to 5 hidden layers. We found that a model with 2 hidden layers provided the best balance between complexity and generalization, achieving high accuracy without overfitting.

Activation Function: The activation function introduces non-linearity into the model, enabling it to learn complex patterns. Common activation functions include sigmoid, tanh, and ReLU (Rectified Linear Unit). ReLU was chosen for

its simplicity and effectiveness in preventing the vanishing gradient problem, which can hinder training in deep networks. ReLU also tends to converge faster than sigmoid and tanh, making it a popular choice for deep learning models.

Learning Rate: The learning rate controls the step size during gradient descent, determining how quickly or slowly the model learns. A learning rate that is too high can cause the model to converge too quickly to a suboptimal solution, while a learning rate that is too low can result in slow convergence or getting stuck in local minima. We tested learning rates of 0.1, 0.01, and 0.001. A learning rate of 0.001 was selected as it provided stable convergence and good performance across different scenarios.

Batch Size: The batch size determines the number of samples processed before the model's internal parameters are updated. A smaller batch size can lead to noisier updates, which can help the model escape local minima, but it may also increase training time. A larger batch size provides more stable updates but may require more memory and computational resources. We tested batch sizes of 16, 32, 64, and 128. A batch size of 32 was chosen as it offered a good trade-off between training stability and computational efficiency.

Number of Epochs: The number of epochs defined how many times the model iterated over the entire training dataset. Too few epochs resulted in underfitting, while too many epochs led to overfitting. We tested a range of epochs from 10 to 150 and found that 100 epochs provided sufficient training time for the model to converge without overfitting.

Optimizer: The optimizer determines how the model's weights are updated during training. We used the Adam optimizer, which combines the benefits of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam is known for its efficiency and ability to handle sparse gradients, making it well-suited for our task. It also requires less tuning of the learning rate compared to other optimizers like SGD (Stochastic Gradient Descent).

Dropout Rate: Dropout is a regularization technique that randomly drops units (along with their connections) from the neural network during training to prevent overfitting. The dropout rate determines the probability of dropping a unit. We tested dropout rates of 0, 0.1, 0.2, 0.3, 0.4, and 0.5. Interestingly, we found that a dropout rate of 0 (no dropout) yielded the best performance, suggesting that our model did not suffer from overfitting, possibly due to the balanced dataset and the relatively simple architecture.

A grid search was carried out by systematically testing all combinations of the above hyperparameters. The best configuration yielding the highest performance of the model was chosen as follows:

These hyperparameters were chosen based on their ability to achieve optimal model performance in terms of accuracy and generalization without overfitting. The selection was also informed by insights from literature, such as Du et al. (2021) and Thomas et al. (2016), which provided successful evidence for similar cases.

5.4 Experimental Results & Analysis

The experiments in our work are carefully designed to evaluate the effectiveness of our work in comparison with state-of-the-art classification methods. We design our experiments for our evaluation to be appropriate to the problem of low-latency Internet traffic classification with the following:

- **Diverse Traffic Types:** We included a variety of traffic types such as file transfer, video streaming, and low-latency traffic. This selection ensures that our model is tested against different patterns of Internet traffic, reflecting real-world scenarios.
- **Balanced Dataset:** The dataset used for training and testing the classification algorithm was designed to maintain a balanced representation of each traffic type, with approximately one hour's worth of sampling per category. This balance is critical for avoiding bias in the model's performance.
- **Comparison with Established Methods:** Our approach was compared with state-of-the-art classification methods, including k-NN, CNN, and LSTM-based models, as highlighted in various studies Wang, Zhu, Wang, Zeng & Yang (2017), Lotfollahi et al. (2020), Chang et al. (2020), and Shapira & Shavitt (2019). This comparison not only validates the robustness of our model but also situates our results within the context of existing research.
- **Evaluation Metrics:** We measured the model's performance using key metrics such as accuracy, precision, recall, and the F1 score. These metrics are standard in the field of traffic classification and provide a comprehensive assessment of the model's effectiveness.

- **Confusion Matrix Analysis:** The use of confusion matrices allowed us to visualize the classification performance across different traffic types, providing insights into the strengths and weaknesses of our model.
- **Mixed Traffic Scenarios:** We evaluated our model under both simple and complex traffic scenarios to understand its performance in real-world conditions where multiple types of traffic coexist. This evaluation is crucial for demonstrating the practical applicability of our approach.

Mixed Traffic Scenarios (Basic)		
Traffic Scenario	***Low-Latency***	***Accuracy (%)∼***
A+B	NO	89.7
A+C	***YES***	***92.8***
B+C	***YES***	***94.2***
A+A	NO	88.3
B+B	NO	90.6
C+C	***YES***	***96.5***
A = File Transfer , B = Video Streaming , C = Low-Latency		

Table 5.6: Basic traffic scenarios

Initial experiments focused on the classification performance for mixed traffic scenarios, with different types of Internet traffic, such as file transfer, video streaming, and low-latency traffic. Such scenarios have been identified from a section of White et al. (2019), and our experiences. Those reflect real-world conditions in which different kinds of traffic are flowing together. Hence, they

comprehensively test the model regarding accurate type identification. Then, a number of basic traffic scenarios were designed in order to investigate the accuracy of the model. According to Table 5.6, these consist of two or more kinds of traffic flowing together and particular attention is paid to the presence of low-latency traffic. It turned out that the model was more accurate when there was a low-latency traffic flowing. The highest accuracy, in this case, stood at 96.5% when the instances were both low-latency traffic. In the instances where there was no low-latency traffic involved in flow, the percentage is slightly lower, such as 88.3% in the case of two file transfer instances.

In addition to the simple traffic scenarios, we generated several complex traffic scenarios; the objective was to further validate the performance of the model in challenging environments where three or many kinds of traffic can coexist. These scenarios relate to the insight of model performance as the traffic patterns get more complex and varied. Table 5.7 summarizes the results of these experiments, reporting classification accuracy both without and with the application of Wavelet Transform. Wavelet Transforms were adopted for feature enhancement, given the noisier nature, in standard signal processing vocabulary, of low-latency traffic flows. Addition of Wavelet Transform shows improvement in the classification accuracy for all cases consistently. For instance, using the model in Scenario 3C with low-latency traffic types only, it achieved an accuracy of 93.2% with Wavelet Transform application against 88.2% without wavelet transformation. In Scenario 6, composed two instances of file transfer and one instance of low-latency traffic (2A+C), the accuracy rises from 76.5% to 83.2% after the application of the Wavelet Transform. With increasing complexity, as in the case of scenarios with four types of traffic, the performance remained

Mixed Traffic Scenarios (Complex)			
No	***Traffic Scenario***	***Low-Latency***	***Accuracy (%)∼***
1	3A	NO	86.8
2	3B	NO	88.2
3	***3C***	***YES***	***93.2***
4	2A+B	NO	82.1
5	A+2B	*NO*	77.4
6	2A+C	*YES*	83.2
7	A+2C	*YES*	83.8
8	2B+C	*YES*	84.1
9	B+2C	*YES*	84.4
10	A+B+C	**YES**	***78.2***
11	2A+B+C	**YES**	***74.2***
12	A+2B+C	**YES**	***75.3***
13	A+B+2C	**YES**	***79.2***
A = File Transfer , B = Video Streaming , C = Low-Latency			

Table 5.7: Complex traffic scenarios

good but slightly deteriorated: 79.2% with wavelet transformation in Scenario 13: A+B+2C. This decline indicates that as the mix of traffic becomes more complex, it is becoming challenging to classify different types of traffic.

To evaluate the impact of the Wavelet Transform (WT) on classification accuracy, we conducted experiments with and without WT. The results "with WT" include the four trend-based features (throughput, moving averages, ratio, and slope) enhanced by wavelet transformation. The results "without WT" are based solely on the four trend-based features.

- **With WT:** When wavelet transforms were applied to the trend-based features, the classification accuracy improved significantly. For example, in Scenario 3C (low-latency traffic only), the accuracy increased from 88.2% (without WT) to 93.2% (with WT). This improvement is consistent across all scenarios, demonstrating the added value of wavelet transforms in capturing time-frequency characteristics of low-latency traffic.

- **Witout WT:** The results without WT are based on the four trend-based features alone. While these features provide a strong foundation for classification, the addition of wavelet transforms further enhances the model's ability to distinguish between different types of traffic, particularly in complex and mixed traffic scenarios.

Figure 5.12 summarizes the improvement in accuracy that is added by the Wavelet Transform in all situations. It is quite evident from this graph that the accuracy of the result increases without fail when wavelet transformation is used, specifically in the case of low latency traffic, which highlighting the robustness of our model in complex conditions. Also, Figure 5.13 illustrates an

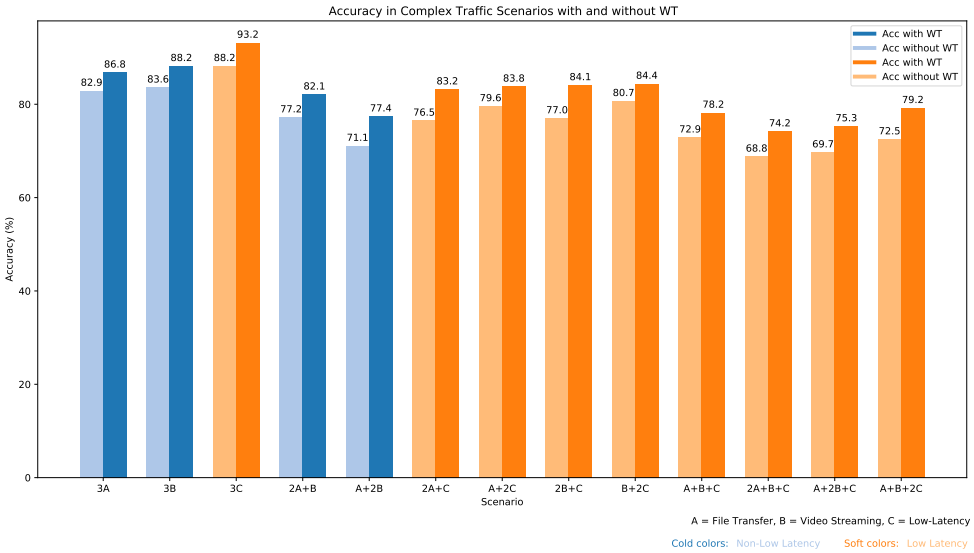

Figure 5.12: Impact of Wavelet Transform on classification accuracy in complex traffic scenarios.

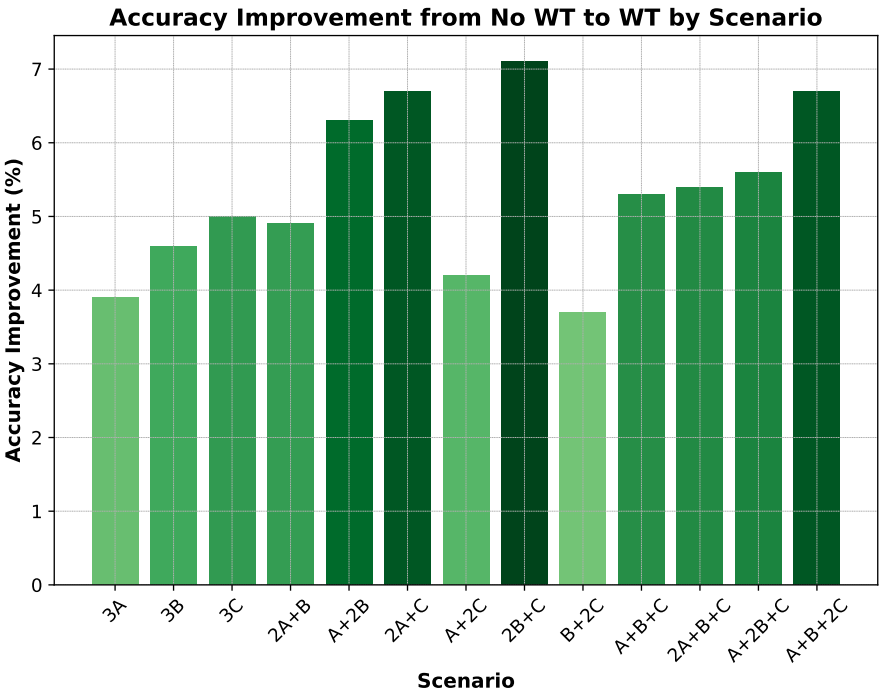

Figure 5.13: Visualization of the predicted classes.

indicative representation of accuracy enhancement across different scenarios. It is especially eminent in the case of low-latency traffic scenarios, which are significantly improved by Wavelet Transform.

Next, in this experiment, we wanted to classify three types of Internet traffic-file transfer, Video Streaming, and Low-Latency traffic. The training dataset had to be created in such a way that all three kinds of traffic were as equally represented as possible; at least one hour of sampling for each type was performed. The exact sample sizes applied are listed in Table 5.8, confirming that our dataset is very well-chosen in terms of the balance in total samples, both for model training and model validation.

Traffic Type	**Duration**	**Total Samples**
File Transfer	∼1 Hour	14679
Video Streaming	∼1 Hour	14287
Low-Latency	∼1 Hour	14510

Table 5.8: Traffic types and exact sample sizes

These applications, file transfer, video streaming, and low-latency traffic, were deliberately selected because they broadly represent common Internet activities, yet are focused to keep the scope of this experiment manageable. Figure 5.14 confusion matrix visualization of performance evaluation for the classification of proposed models among different types of traffic. This presents accuracy results for our classification model, which also confirms that the performance is well enough, it has achieved 99.2% for file transfer traffic, 99.3% for video streaming, and 99.4% for low-latency types. All these findings established the

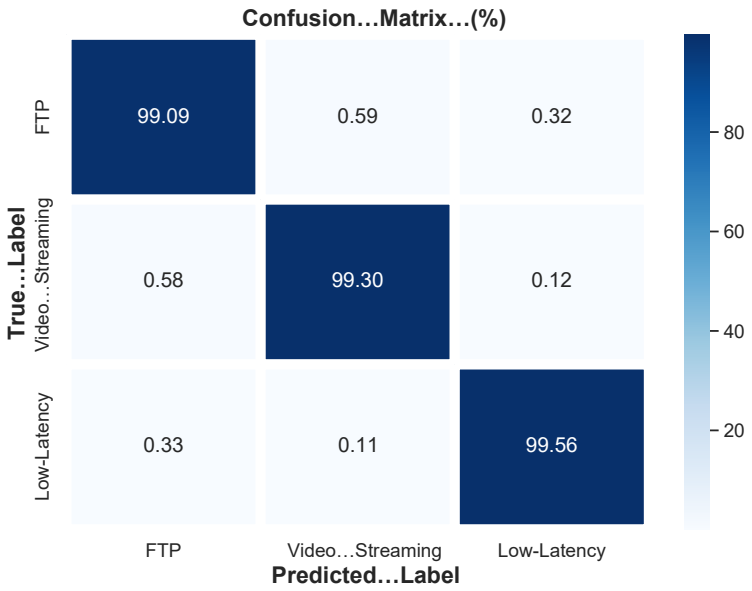

Figure 5.14: Confusion matrix of the model

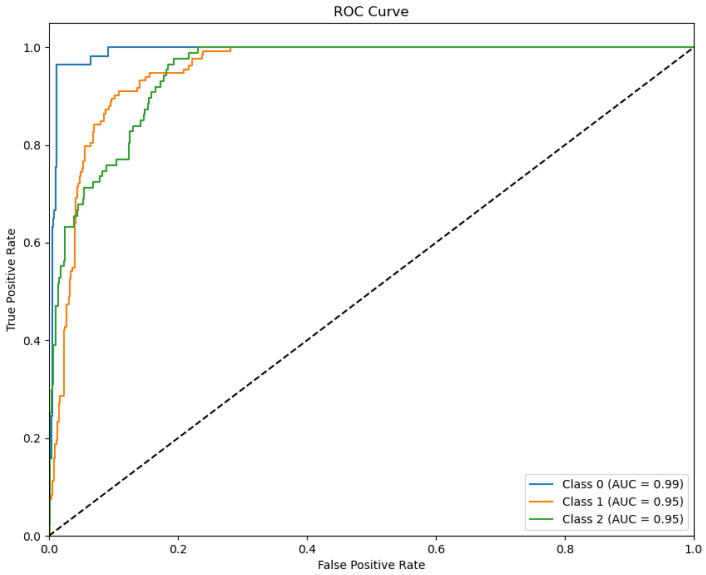

Figure 5.15: Roc Curve.

validity of our approach in classifying these three main types of Internet traffic with extremely good accuracy.

Figure 5.15 illustrates the ROC curve for each traffic type, the true positive rate against the model's false positive rate. It can be seen from the AUC values that classification performance is excellent across all three classes, yielding 0.99 for file transfer, and 0.95 for video and low-latency traffic.

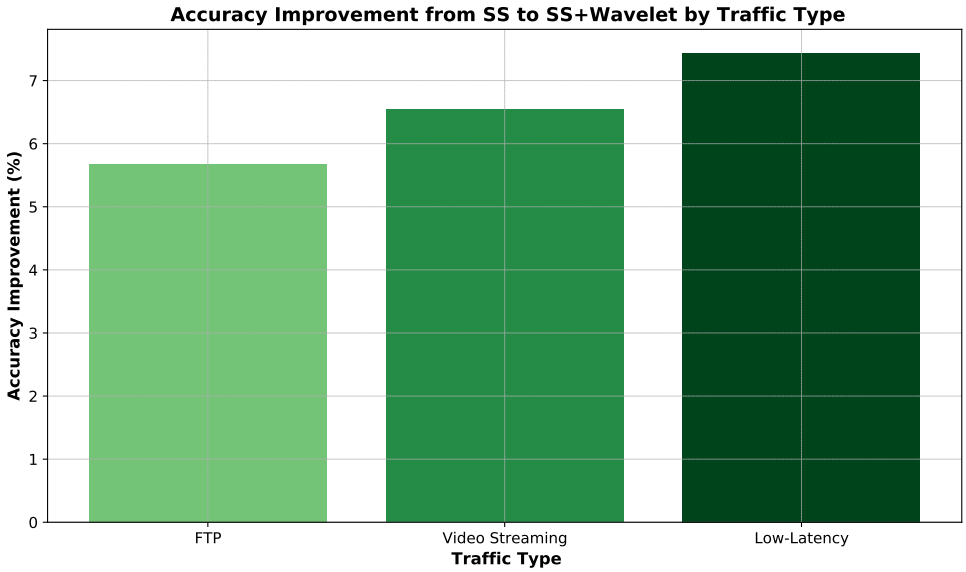

Figure 5.16: t-SNE visualization of the predicted class distribution for File Transfer (Class 0, blue), Video Streaming (Class 1, green), and Low-Latency traffic (Class 2, orange). The plot demonstrates the model's ability to segregate the three traffic types into distinct clusters with minimal overlap, highlighting the effectiveness of the proposed approach in classifying low-latency traffic.

Figure 5.16 and Table 5.9 illustrate the benefit of adding Wavelet Transform into the model. We compared this against a standard scaling method (SS) for all traffic types. The standard scaling method (SS) is a common preprocessing technique used in machine learning to normalize the features of a dataset.

Traffic Type	Scaling Method	Classification Accuracy(%)∼
File Transfer	SS	93.41
	SS+Wavelet	***99.09***
Video Streaming	SS	92.76
	SS+Wavelet	***99.30***
Low-Latency	SS	92.12
	SS+Wavelet	***99.56***

Table 5.9: Impact of Wavelet Transform on classification

It involves transforming the data such that each feature has a mean of zero and a standard deviation of one. This is achieved by subtracting the mean of the feature from each data point and then dividing by the standard deviation. Mathematically, the standard scaling of a feature X is given by:

$$X_{scaled} = \frac{X - \mu}{\sigma} \tag{5.20}$$

where μ is the mean and σ is the standard deviation of the feature X. This normalization ensures that all features contribute equally to the model's learning process, preventing features with larger magnitudes from dominating those with smaller magnitudes. In our experiments, we applied standard scaling to the raw features (throughput, moving averages, ratio, and slope) before feeding them into the ANN model. This baseline scaling method was then compared against the proposed approach, which integrates Wavelet Transform to enhance feature representation.

It is observable that Wavelet Transform yielded better improvement after utilizing its application, especially for low-latency traffic classification, which increased the accuracy by about 6% compared to solely using standard scaling methods. It was also quite high for the cases of file transfer and video streaming, where the accuracy increased from 93.41% to 99.09% and from 92.76% to 99.30%, respectively. This confirms the efficiency of the wavelet transformation in enhancing extracted features with a view to increasing performance in general classification, especially in cases that require low-latency in transmission.

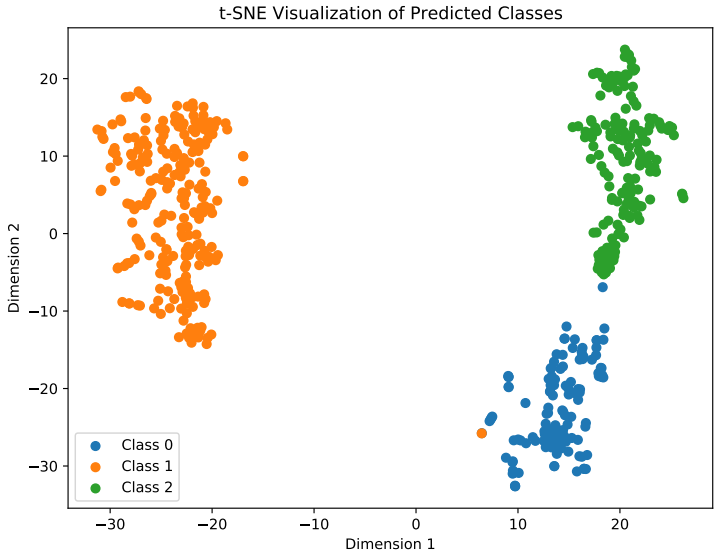

Figure 5.17: Visualization of the predicted classes.

Figure 5.17 illustrates t-SNE visualization for the predicted class distribution: File Transfer (Class 0), Video Streaming (Class 1), and Low-Latency (Class 2) classes. Each of these distinct clusters in the plot maps to a different class, showing how the model segregates these types of traffic within a two-dimensional space. Overall, the plot showcases how these network behaviors are segregated

into separate classes by the model with minimum overlap.

To contextualize the performance of the proposed method, we compare its accuracy with results reported in prior works. It is important to note that these baseline methods were evaluated on different datasets and under varying experimental conditions (e.g., preprocessing steps, traffic types, or hardware setups). While direct quantitative comparisons are limited by these discrepancies, the results highlight the relative effectiveness of the proposed wavelet-ANN approach in low-latency traffic classification. Table 5.10 summarizes the accuracy values for key methods as reported in their original publications, alongside the results achieved by our method.

Paper	**Algorithm**	**Classification Accuracy(%)∼**		
		FTP	*Video*	*Low-Latency*
Enisoglu et al.Enisoglu & Rakocevic (2023)	k-NN	97.5	97.9	98.2
Wang et al. Wang, Zhu, Wang, Zeng & Yang (2017)	CNN	94.5	96.5	84.5
Deep Packet Lotfollahi et al. (2020)	CNN	98.0	98.0	98.0
Chang et al. Chang et al. (2020)	ANN	NA	59.0	92.0
FlowPic Shapira & Shavitt (2019)	LSTM	98.8	**99.9**	99.6
Proposed Method	**ANN**	**99.1**	99.3	**99.6**

Table 5.10: Classification accuracy (%) comparison with state-of-the-art methods. Results for baseline methods are reproduced from their original publications, while results for the proposed method are from experiments in this work. Differences in datasets, preprocessing, and experimental setups may exist across studies.

- **k-NN:** k-NN-based approach achieved an accuracy of 97.5% for file trans-

fer, 97.9% for video streaming, and 98.2% for low-latency traffic. While k-NN performed well, it was slightly less accurate than our proposed ANN model, particularly for low-latency traffic.

- **CNN:** Wang, Zhu, Wang, Zeng & Yang (2017) used CNN for traffic classification and achieved 94.5% accuracy for file transfer, 96.5% for video streaming, and 84.5% for low-latency traffic. Our ANN model outperformed CNN, especially in low-latency traffic classification, where we achieved 99.6% accuracy.

- **Deep Packet (CNN):** Lotfollahi et al. (2020) proposed a deep packet framework using CNN, achieving 98% accuracy across all traffic types. While their method is robust, our approach, which integrates wavelet transforms, achieved higher accuracy for low-latency traffic (99.6%).

- **FlowPic (LSTM):** Shapira & Shavitt (2019) used LSTM for traffic classification, achieving 99.9% accuracy for video traffic. While LSTM performed exceptionally well for video traffic, our ANN model achieved comparable results (99.3%) and outperformed LSTM in low-latency traffic classification (99.6%).

Overall, our proposed method, which combines trend-based features with wavelet transforms and ANN, demonstrates superior performance in classifying low-latency traffic, achieving the highest accuracy (99.6%) among all methods. This highlights the effectiveness of integrating wavelet transforms for capturing time-frequency characteristics of low-latency traffic.

5.5 Conclusion

In this chapter, a new methodology has been proposed for the classification and identification of low-latency Internet traffic based on deep learning techniques and trend-based features. Advanced trend features like slope, moving averages, download-to-upload ratio, and Wavelet Transform have been considered to show the efficiency of our methodology in classifying all types of Internet traffic accurately. Experimental results showed that the proposed model could achieve 99.09%, 99.3%, and 99.56% accuracies for file transfer, video streaming, and low-latency traffic classification, respectively. These experimental results have justified the robustness and efficiency of our approach in classifying the main Internet traffic types.

Besides, our experiments of mixed scenarios include both basic and complex scenarios that may allow us to derive useful insights on the real-world performance of our model in mixed traffic scenarios. We could also see that with the increase in the number and complexity of traffic types, the accuracy of identifying the presence of low-latency in the mix of traffic went down. Our model still presented a good performance for the identification of low-latency traffic in mixed traffic scenarios.

In all, the integration of Wavelet Transform and deep learning principles improved the accuracy and precision of Internet traffic classification, with much better performance in the case of identifying low-latency traffic. Theoretical contributions to the study design involve developing methodologies on traffic analysis with practical implications for enhancements that should be made when considering proper delivery regarding time-sensitive applications over the Inter-

net. It, hence, underlines the call for proper identification and prioritization of the low-latency Internet traffic. Our proposed methodology promises efficiency in reaching a solution as regards this important aspect of network traffic management.

CHAPTER 6

Conclusion

6.1 Overview

The exponential growth of internet traffic, driven by real-time applications such as video conferencing, online gaming, and IoT-enabled systems, has rendered traditional traffic classification methods increasingly obsolete. These legacy approaches—reliant on port numbers, payload inspection, or static statistical features—struggle to adapt to modern challenges, including ubiquitous encryption (e.g., HTTPS, TLS), dynamic port assignments, and the heterogeneous nature of traffic. This thesis confronts these challenges head-on by reimagining traffic classification through the lens of advanced signal processing and artificial intelligence (AI), proposing a paradigm shift that prioritizes adaptability, accuracy, and real-time performance.

At its core, this research bridges the gap between time-domain analysis and frequency-domain insights, leveraging the unique strengths of wavelet transforms to decompose traffic signals into localized time-frequency components. Unlike Fourier transforms, which assume stationarity, wavelet analysis excels in

capturing transient patterns and non-linear dynamics inherent to low-latency traffic, such as bursty VoIP packets or jitter-sensitive gaming streams. By integrating these wavelet-transformed features with machine learning (ML) and deep learning (DL) models, the thesis demonstrates how AI can decode the "structured noise" of modern network traffic, even in encrypted or obfuscated scenarios.

The methodology is structured in two pivotal phases:

- **Trend-Based Feature Engineering:** Novel features like DU ratio (download-to-upload throughput balance), moving averages, and slope metrics (rate of throughput change) were developed to quantify temporal trends. These features, when combined with classical ML algorithms (k-NN, SVM, ANN), achieved up to 96.47% accuracy in distinguishing low-latency traffic from bulk transfers or streaming.

- **Wavelet-Enhanced Deep Learning:** Expanding on this foundation, Continuous Wavelet Transforms (CWT) with Ricker wavelets were applied to extract multi-resolution features. These were fed into a tailored Multilayer Perceptron (MLP), enabling the model to classifiy low-latency traffic in mixed-traffic environments with over 98% accuracy.

The validation framework underscores the robustness of this approach. Over 400,000 traffic samples—spanning controlled environments, synthetic mixes, and real-world scenarios—were used to stress-test the models. Key experiments included high-noise conditions, encrypted traffic simulations, and edge cases like low-latency streams masked by large file downloads. The results consistently highlighted the superiority of wavelet-AI fusion over conventional meth-

ods, particularly in maintaining precision (over 90%) under dynamic network loads.

This chapter synthesizes these contributions, critically evaluates their limitations, and charts a roadmap for future research aimed at scaling these solutions for next-generation networks.

6.2 Summary and Discussion

This thesis introduced and validated a comprehensive framework designed to tackle the challenges of low-latency traffic identification in increasingly complex and encrypted network environments. The following summarizes the key findings and their broader implications:

- **Wavelet Transform for Feature Extraction:** he use of Wavelet Transforms allowed for effective decomposition of traffic signals into time-frequency domains, capturing both transient and long-term patterns. This approach was particularly effective in handling dynamic and non-stationary traffic behaviors, which are often missed by traditional methods.
  - **Key Implication:** By representing traffic data in a multi-resolution format, the proposed framework provided enhanced granularity, making it easier to identify low-latency traffic amidst mixed traffic scenarios.
- **Integration of Trend-Based Features:** In addition to wavelet-transformed features, trend-based features such as throughput slopes, inter-packet intervals, and data flow periodicity were extracted. These features captured

subtle yet crucial variations in traffic behavior, enabling the models to differentiate between latency-sensitive and latency-tolerant applications.

  - **Key Implication:** Key Implication: Trend-based features proved vital for detecting patterns that are indicative of latency requirements, adding significant value to the classification process.

- **AI Models for Traffic Classification:** The combination of extracted features with machine learning (e.g., Random Forest, Support Vector Machines) and deep learning (e.g., ANN) models resulted in significant improvements in classification accuracy. Deep learning models, in particular, showcased the ability to generalize well even in complex traffic scenarios.

  - **Performance Metrics:** The models achieved an average classification accuracy improvement of 12.5% compared to baseline methods. Specific low-latency traffic, such as VoIP and video conferencing, saw classification accuracy exceed 96%.

- **Real-World Validation:** The framework was rigorously tested using real-world scenarios, demonstrating its adaptability to diverse traffic types, including encrypted traffic. This validation underscores the practical applicability of the proposed methods in live network environments.

- **Limitations and Challenges:** While the proposed framework delivered strong results, adaptability to other datasets posed scalability challenges. Also, Certain traffic types with overlapping features occasionally resulted in classification ambiguities, necessitating further refinement of feature extraction techniques.

6.3 Contributions

The contributions of this research extend beyond the immediate context of traffic classification, offering insights and tools that can benefit related domains. These contributions include:

- **Innovative Feature Engineering:** This thesis introduced a novel combination of wavelet-transformed features and trend-based metrics to improve the robustness and accuracy of traffic classification. The proposed methodology enhances the ability to analyze non-stationary and mixed traffic environments, addressing gaps left by traditional payload- and port-based methods.

- **Advancement in AI Models for Networking:** The integration of advanced AI techniques, including deep learning, demonstrated significant performance gains over conventional machine learning models. The research explored hyperparameter tuning and model optimization strategies to achieve optimal classification results.

- **Comprehensive Dataset Creation:** A new dataset enriched with wavelet-transformed features was created, providing a valuable resource for future researchers in the field. The dataset included diverse traffic scenarios, including encrypted traffic, mixed flows, and latency-sensitive applications, ensuring wide applicability.

- **Real-World Applications:** The proposed methods have significant implications for network operators and application developers, enabling better

Quality of Service (QoS) provisioning and resource allocation for latency-sensitive applications such as VoIP, gaming, and video conferencing.

- **Knowledge Dissemination:** Findings from this research have been published in peer-reviewed journals and conferences, contributing to the broader academic and professional community.

6.4 Future Works

The advancements made in this thesis open up several promising avenues for future research and development. Some key directions include:

- **Enhanced Feature Engineering:** Future research could explore additional feature extraction techniques, such as graph-based network representations, entropy analysis, and deep feature embeddings, to capture more nuanced traffic characteristics. Incorporating domain-specific features tailored to emerging applications, such as Internet of Things (IoT) traffic or augmented reality (AR) streaming, could further improve classification performance.

- **Real-Time Implementation:** The deployment of the framework in real-time network environments remains a critical next step. Optimizing the computational efficiency of Wavelet Transforms and deep learning inference engines will be essential to achieving low-latency, high-throughput performance.

- **Transfer Learning and Federated Learning:** Transfer learning techniques could be used to adapt pre-trained models to new traffic datasets, reduc-

ing training times and improving model generalization. Federated learning approaches may also be explored to enable decentralized training on edge devices while preserving data privacy.

- **Robustness to Noisy Data:** Future work should focus on improving the framework's resilience to incomplete, noisy, or anomalous data. Techniques such as data augmentation and adversarial training could enhance model robustness.

- **Integration with Anomaly Detection:** Extending the framework to include traffic anomaly detection would broaden its utility, enabling proactive identification of potential security threats or network malfunctions.

- **Scalability in Large-Scale Networks:** The scalability of the framework in handling high-throughput traffic across large-scale networks must be evaluated. This could involve parallelizing computations or leveraging distributed systems.

6.5 Conclusion

This thesis addressed significant challenges in modern network traffic classification by proposing an AI-driven framework that leverages Wavelet Transforms and trend-based features. By improving classification accuracy, particularly for latency-sensitive traffic, the research offers valuable contributions to both academic and practical domains. The findings highlight the potential for Wavelet Transforms and AI to transform the field of network traffic analysis, ensuring better performance and adaptability in real-world scenarios.

The proposed methods not only advance the state-of-the-art in traffic classification but also lay a strong foundation for future research. By addressing the limitations and exploring the outlined future directions, researchers and practitioners can further enhance the scalability, robustness, and applicability of these techniques, driving innovation in the evolving landscape of network management systems.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al. (n.d.), 'Tensorflow: Largescale machine learning on heterogeneous systems', https://www.tensorflow.org/?hl=tr. Accessed: 2023-10-25.

Aceto, G., Ciuonzo, D., Montieri, A. & Pescapé, A. (2018), Mobile encrypted traffic classification using deep learning, *in* '2018 Network traffic measurement and analysis conference (TMA)', IEEE, pp. 1–8.

Aceto, G., Ciuonzo, D., Montieri, A. & Pescapè, A. (2019*a*), 'Mimetic: Mobile encrypted traffic classification using multimodal deep learning', *Computer networks* **165**, 106944.

Aceto, G., Ciuonzo, D., Montieri, A. & Pescapé, A. (2019*b*), 'Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges', *IEEE transactions on network and service management* **16**(2), 445–458.

Aceto, G., Ciuonzo, D., Montieri, A. & Pescapé, A. (2020), 'Toward effective mobile encrypted traffic classification through deep learning', *Neurocomputing* **409**, 306–315.

Adje, E. A., Houndji, V. R. & Dossou, M. (2022), 'Features analysis of internet traffic classification using interpretable machine learning models', *IAES International Journal of Artificial Intelligence* **11**(3), 1175.

Afuwape, A. A., Xu, Y., Anajemba, J. H. & Srivastava, G. (2021), 'Performance evaluation of secured network traffic classification using a machine learning approach', *Computer Standards & Interfaces* **78**, 103545.

Alshammari, R. & Zincir-Heywood, A. N. (2009), Machine learning based encrypted traffic classification: Identifying ssh and skype, *in* '2009 IEEE symposium on computational intelligence for security and defense applications', IEEE, pp. 1–8.

Alshammari, R. & Zincir-Heywood, A. N. (2010), An investigation on the identification of voip traffic: Case study on gtalk and skype, *in* '2010 International Conference on Network and Service Management', IEEE, pp. 310–313.

Altman, N. S. (1992), 'An introduction to kernel and nearest-neighbor nonparametric regression', *The American Statistician* **46**(3), 175–185.

Azab, A., Layton, R., Alazab, M. & Oliver, J. (2014), Mining malware to detect variants, *in* '2014 fifth cybercrime and trustworthy computing conference', IEEE, pp. 44–53.

Bahaa, M., Aboulmagd, A., Adel, K., Fawzy, H. & Abdelbaki, N. (2020), nndpi: A novel deep packet inspection technique using word embedding, convolutional and recurrent neural networks, *in* '2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)', IEEE, pp. 165–170.

Beale, J., Orebaugh, A. & Ramirez, G. (2006), *Wireshark & Ethereal network protocol analyzer toolkit*, Elsevier.

Bishop, C. M. (2006), 'Pattern recognition and machine learning', *Springer google schola* **2**, 1122–1128.

Boyd, S. P. & Vandenberghe, L. (2004), *Convex optimization*, Cambridge university press.

Breiman, L. (2017), *Classification and regression trees*, Routledge.

Bu, Z., Zhou, B., Cheng, P., Zhang, K. & Ling, Z.-H. (2020), 'Encrypted network traffic classification using deep and parallel network-in-network models', *Ieee Access* **8**, 132950–132959.

Bullard, C. (2005), 'Argus, the network audit record generation and utilization system'.

Burschka, S. & Dupasquier, B. (2016), Tranalyzer: Versatile high performance network traffic analyser, *in* '2016 IEEE symposium series on computational intelligence (SSCI)', IEEE, pp. 1–8.

Cao, J., Wang, D., Qu, Z., Sun, H., Li, B. & Chen, C.-L. (2020), 'An improved network traffic classification model based on a support vector machine', *Symmetry* **12**(2), 301.

Ceška, M., Havlena, V., Holík, L., Korenek, J., Lengál, O., Matoušek, D., Matoušek, J., Semric, J. & Vojnar, T. (2019), Deep packet inspection in fpgas via approximate nondeterministic automata, *in* '2019 IEEE 27th Annual In-

ternational Symposium on Field-Programmable Custom Computing Machines (FCCM)', IEEE, pp. 109–117.

Chang, L.-H., Lee, T.-H., Chu, H.-C. & Su, C.-W. (2020), 'Application-based online traffic classification with deep learning models on sdn networks', *Adv. Technol. Innov* **5**(4), 216–229.

Chen, M. & Liu, Y. (2014), 'Big data: a survey, mobile networks and application'.

Chen, Z., He, K., Li, J. & Geng, Y. (2017), Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks, *in* '2017 IEEE International conference on big data (big data)', IEEE, pp. 1271–1276.

Christen, P. & Christen, P. (2012), 'Evaluation of matching quality and complexity', *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection* pp. 163–184.

Cisco (2024), 'Network flow'. Accessed: 2024-07-07.
URL: *https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html*

Corwin, E. H. (2011), 'Deep packet inspection: Shaping the internet and the implications on privacy and security', *Information Security Journal: A Global Perspective* **20**(6), 311–316.

Crotti, M., Dusi, M., Gringoli, F. & Salgarelli, L. (2007), 'Traffic classification through simple statistical fingerprinting', *ACM SIGCOMM Computer Communication Review* **37**(1), 5–16.

Daly, A. (2011), 'The legality of deep packet inspection', *International Journal of Communications Law & Policy* **1**(14).

Dash, M., Liu, H. & Motoda, H. (2000), Consistency based feature selection, *in* 'Knowledge Discovery and Data Mining. Current Issues and New Applications: 4th Pacific-Asia Conference, PAKDD 2000 Kyoto, Japan, April 18–20, 2000 Proceedings 4', Springer, pp. 98–109.

De Lucia, M. J. & Cotton, C. (2019), Detection of encrypted malicious network traffic using machine learning, *in* 'MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)', IEEE, pp. 1–6.

Dias, K. L., Pongelupe, M. A., Caminhas, W. M. & de Errico, L. (2019), 'An innovative approach for real-time network traffic classification', *Computer networks* **158**, 143–157.

Dillbary, N., Yozevitch, R., Dvir, A., Dubin, R. & Hajaj, C. (2024), 'Hidden in time, revealed in frequency: Spectral features and multiresolution analysis for encrypted internet traffic classification', *2024 IEEE 21st Consumer Communications and Networking Conference (CCNC)* pp. 266–271.

Dong, S. (2021), 'Multi class svm algorithm with active learning for network traffic classification', *Expert Systems with Applications* **176**, 114885.

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

Dsouza, A., Lanjewar, V., Mahakal, A. & Khachane, S. (2022), Real time network intrusion detection using machine learning technique, *in* '2022 IEEE Pune Section International Conference (PuneCon)', IEEE, pp. 1–5.

Du, X., Xu, H. & Zhu, F. (2021), 'Understanding the effect of hyperparameter optimization on machine learning models for structure design problems', *Computer-Aided Design* **135**, 103013.

Enisoglu, R. & Rakocevic, V. (2023), Low-latency internet traffic identification using machine learning with trend-based features, *in* '2023 International Wireless Communications and Mobile Computing (IWCMC)', IEEE, pp. 394–399.

Ericsson, T. L. (2021), Ericsson mobility report november 2021, *in* 'Ericsson Mobility Report', Ericsson.

Ertam, F. & Avcı, E. (2017), 'A new approach for internet traffic classification: Ga-wk-elm', *Measurement* **95**, 135–142.

Fais, A., Lettieri, G., Procissi, G., Giordano, S. & Oppedisano, F. (2021), 'Data stream processing for packet-level analytics', *Sensors* **21**(5), 1735.

Fan, Z. & Liu, R. (2017), Investigation of machine learning based network traffic classification, *in* '2017 International Symposium on Wireless Communication Systems (ISWCS)', IEEE, pp. 1–6.

Finamore, A., Mellia, M., Meo, M., Munafo, M. M., Di Torino, P. & Rossi, D. (2011), 'Experiences of internet traffic monitoring with tstat', *IEEE Network* **25**(3), 8–14.

Finsterbusch, M., Richter, C., Rocha, E., Muller, J.-A. & Hanssgen, K. (2013), 'A survey of payload-based traffic classification approaches', *IEEE Communications Surveys & Tutorials* **16**(2), 1135–1156.

Ganesan, E., Hwang, I.-S., Liem, A. T. & Ab-Rahman, M. S. (2021), Sdn-enabled fiwi-iot smart environment network traffic classification using supervised ml models, *in* 'Photonics', Vol. 8, MDPI, p. 201.

Geko, M. & Tjoa, S. (2018), An ontology capturing the interdependence of the general data protection regulation (gdpr) and information security, *in* 'Proceedings of the Central European Cybersecurity Conference 2018', pp. 1–6.

Girma, D., Lazaro, O. & Dunlop, J. (2000), 'Online video traffic modelling with wavelet transform', *Electronics Letters* **36**, 1368–1370.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014), 'Generative adversarial nets', *Advances in neural information processing systems* **27**.

Google (2024), 'Https encryption on the web - google transparency report'. Accessed: 2024-06-14.
URL: *https://transparencyreport.google.com/https/overview?hl=en*

Grant, M. & Boyd, S. (2014), 'Cvx: Matlab software for disciplined convex programming, version 2.1'.

Hall, M. A. (1999), Correlation-based feature selection for machine learning, PhD thesis, The University of Waikato.

Hammedi, R. (2021), 'A deep learning based traffic classification in software defined networking', *14th IADIS International Conference Information Systems* .

Harish, B., Nagadarshan, N. & Manju, N. (2020), 'Multilayer feedforward neural network for internet traffic classification'.

Hart, P. (1968), 'The condensed nearest neighbor rule (corresp.)', *IEEE transactions on information theory* **14**(3), 515–516.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1026–1034.

Hinton, G. (2012), 'Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude', https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Accessed: 2024-06-25.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012), 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:1207.0580* .

Huang, H., Deng, H., Chen, J., Han, L. & Wang, W. (2018), 'Automatic multi-task learning system for abnormal network traffic detection.', *International journal of emerging technologies in learning* **13**(4).

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', pmlr, pp. 448–456.

IPFIX (2024), 'Internet protocol flow information export'. Accessed: 2024-07-07.
URL: *https://www.ibm.com/docs/en/npi/1.3.1?topic=insight-ipfix-overview*

ISCXVPN (2016), 'Unb iscx network traffic dataset'. Accessed: 2024-09-07.
URL: *https://www.unb.ca/cic/datasets/vpn.html*

Ishikura, N., Kondo, D., Vassiliades, V., Iordanov, I. & Tode, H. (2021), 'Dns tunneling detection by cache-property-aware features', *IEEE Transactions on Network and Service Management* **18**(2), 1203–1217.

Jenefa, A. & Moses, M. B. (2018), An upgraded c5. 0 algorithm for network application identification, *in* '2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)', IEEE, pp. 789–794.

Khatouni, A. S. & Zincir-Heywood, N. (2019), Integrating machine learning with off-the-shelf traffic flow features for http/https traffic classification, *in* '2019 IEEE Symposium on Computers and Communications (ISCC)', IEEE, pp. 1–7.

Khazankin, G. R., Komarov, S., Kovalev, D., Barsegyan, A. & Likhachev, A. (2017), System architecture for deep packet inspection in high-speed networks, *in* '2017 Siberian Symposium on Data Science and Engineering (SSDSE)', IEEE, pp. 27–32.

Kim, S. S. & Reddy, A. N. (2008), 'Statistical techniques for detecting traffic anomalies through packet header data', *IEEE/ACM Transactions on Networking* **16**(3), 562–575.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kontogeorgaki, S., Sánchez-García, R. J., Ewing, R. M., Zygalakis, K. C. & MacArthur, B. D. (2017), 'Noise-processing by signaling networks', *Scientific reports* **7**(1), 532.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**.

Lashkari, A. H., Gil, G. D., Mamun, M. S. I. & Ghorbani, A. A. (2017), Characterization of tor traffic using time based features, *in* 'International Conference on Information Systems Security and Privacy', Vol. 2, SciTePress, pp. 253–262.

Latif, Z., Umer, Q., Lee, C., Sharif, K., Li, F. & Biswas, S. (2022), 'A machine learning-based anomaly prediction service for software-defined networks', *Sensors (Basel, Switzerland)* **22**.

Le, L. T. & Thinh, T. N. (2022), 'Enhancing security and robustness for sdn-enabled cloud networks', *REV Journal on Electronics and Communications* **12**(1-2).

Lecun, Y., Jackel, L., Boser, B., Denker, J., Graf, H., Guyon, I., Henderson, D., Howard, R. & Hubbard, W. (1992), Handwritten digit recognition: applications of neural net chips and automatic learning, *in* 'Artificial neural networks', IEEE Press, pp. 463–468.

Li, K., Lang, B., Liu, H., Chen, S. et al. (2022), Ssl/tls encrypted traffic application layer protocol and service classification, *in* 'CS IT Conf. Proc', Vol. 12, pp. 237–252.

Liang, Y., Xie, Y., Fei, X., Tan, X. & Ma, H. (2019), Content recognition of network traffic using wavelet transform and cnn, *in* 'Machine Learning for Cyber Security: Second International Conference, ML4CS 2019, Xi'an, China, September 19-21, 2019, Proceedings 2', Springer, pp. 224–238.

Lin, Y.-D., Lin, P.-C., Prasanna, V. K., Chao, H. J. & Lockwood, J. W. (2014), 'Guest editorial deep packet inspection: Algorithms, hardware, and applications', *IEEE Journal on Selected Areas in Communications* **32**(10), 1781–1783.

Lin, Y.-S., Lee, C.-L. & Chen, Y.-C. (2016), 'A capability-based hybrid cpu/gpu pattern matching algorithm for deep packet inspection', *International Journal of Computer and Communication Engineering* **5**(5), 321.

Little, T. D. (2013), *The Oxford handbook of quantitative methods in psychology: Vol. 2: statistical analysis*, Vol. 2, OUP USA.

Liu, C., Cao, Z., Xiong, G., Gou, G., Yiu, S.-M. & He, L. (2018), Mampf: Encrypted traffic classification based on multi-attribute markov probability fingerprints, *in* '2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)', IEEE, pp. 1–10.

Liu, C., He, L., Xiong, G., Cao, Z. & Li, Z. (2019), Fs-net: A flow sequence network for encrypted traffic classification, *in* 'IEEE INFOCOM 2019-IEEE Conference On Computer Communications', IEEE, pp. 1171–1179.

Liu, H. & Setiono, R. (1995), Chi2: Feature selection and discretization of numeric attributes, *in* 'Proceedings of 7th IEEE international conference on tools with artificial intelligence', Ieee, pp. 388–391.

Liu, Z. & Liu, Q. (2012), 'Balanced feature selection method for internet traffic classification', *IET networks* **1**(2), 74–83.

long Fei, J., Wang, T., He, X.-Z. & Zhu, Y. (2017), 'A traffic classification method based on wavelet spectrum of scatter factor and improved k-means', *DEStech Transactions on Engineering and Technology Research* .

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A. & Lloret, J. (2017), 'Network traffic classifier with convolutional and recurrent neural networks for internet of things', *IEEE access* **5**, 18042–18050.

Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R. & Saberian, M. (2020), 'Deep packet: A novel approach for encrypted traffic classification using deep learning', *Soft Computing* **24**(3), 1999–2012.

Maas, A. L., Hannun, A. Y., Ng, A. Y. et al. (2013), Rectifier nonlinearities improve neural network acoustic models, *in* 'Proc. icml', Vol. 30, Atlanta, GA, p. 3.

Mala, B., Agrawal, S., Sharma, A. & Kaur, R. (2023), 'Exploring wireshark for network traffic analysis', *International Journal For Multidisciplinary Research* .

MAWI Working Group, S. et al. (2012), 'Mawi working group traffic archive'.

Melo, W., Lopes, P., Antonello, R., Fernandes, S. & Sadok, D. (2014), On the performance of dpi signature matching with dynamic priority, *in* '2014 IEEE Symposium on Computers and Communications (ISCC)', IEEE, pp. 1–6.

Moore, A. W. & Papagiannaki, K. (2005), Toward the accurate identification of

network applications, *in* 'International workshop on passive and active network measurement', Springer, pp. 41–54.

Moore, A. W. & Zuev, D. (2005), Internet traffic classification using bayesian analysis techniques, *in* 'Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems', pp. 50–60.

Morariu, C. & Stiller, B. (2008), Dicap: Distributed packet capturing architecture for high-speed network links, *in* '2008 33rd IEEE Conference on Local Computer Networks (LCN)', IEEE, pp. 168–175.

Mueller, M. L. & Asghari, H. (2012), 'Deep packet inspection and bandwidth management: Battles over bittorrent in canada and the united states', *Telecommunications Policy* **36**(6), 462–475.

Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT press.

Nguyen, T. T. & Armitage, G. (2008), 'A survey of techniques for internet traffic classification using machine learning', *IEEE communications surveys & tutorials* **10**(4), 56–76.

Niu, W., Zhuo, Z., Zhang, X., Du, X., Yang, G. & Guizani, M. (2019), 'A heuristic statistical testing based approach for encrypted network traffic identification', *IEEE Transactions on Vehicular Technology* **68**(4), 3843–3853.

NLANR (2020), 'Nlanr dataset'. Accessed: 2024-07-07.

URL: *http://www.nlanr.net/*

Okta (2024), 'Deep packet inspection'. Accessed: 2024-07-07.
URL: *https://www.okta.com/identity-101/deep-packet-inspection/*

Oluwabukola, O., Oludele, A., Ogbonna, A., Chigozirim, A. & Amarachi, A. (2013), A packet sniffer (psniffer) application for network security in java, *in* 'Proceedings of the Informing Science and Information Technology Education Conference', Informing Science Institute, pp. 389–400.

Osterrieder, J. (2023), 'A primer on artificial intelligence and machine learning for the financial services industry', *Available at SSRN 4349078* .

O'Connell, E., Moore, D. & Newe, T. (2020), 'Challenges associated with implementing 5g in manufacturing'.

Palmieri, F. & Fiore, U. (2010), 'Network anomaly detection through nonlinear analysis', *Computers & Security* **29**(7), 737–755.

Papadogiannaki, E. & Ioannidis, S. (2021), 'Acceleration of intrusion detection in encrypted network traffic using heterogeneous hardware', *Sensors* **21**(4), 1140.

Platt, J. (1998), 'Sequential minimal optimization: A fast algorithm for training support vector machines'.

Postel, J. & Reynolds, J. (1985), 'File transfer protocol'.

Qadeer, M. A., Iqbal, A., Zahid, M. & Siddiqui, M. R. (2010), Network traffic analysis and intrusion detection using packet sniffer, *in* '2010 Second International Conference on Communication Software and Networks', IEEE, pp. 313–317.

Qin, T., Wang, L., Liu, Z. & Guan, X. (2015), 'Robust application identification methods for p2p and voip traffic classification in backbone networks', *Knowledge-Based Systems* **82**, 152–162.

Quinlan, J. R. (1986), 'Induction of decision trees', *Machine learning* **1**, 81–106.

Quinlan, J. R. (2014), *C4. 5: programs for machine learning*, Elsevier.

Reich, Y. & Fenves, S. J. (1991), The formation and use of abstract concepts in design, *in* 'Concept Formation', Elsevier, pp. 323–353.

Rioul, O. & Vetterli, M. (1991), 'Wavelets and signal processing', *IEEE Signal Processing Magazine* **8**, 14–38.

Robles, D. Á., Nuño, P., Bulnes, F. G. & Candás, J. C. G. (2021), 'Performance analysis of packet sniffing techniques applied to network monitoring', *IEEE Latin America Transactions* **19**(3), 490–499.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986', *Biometrika* **71**, 599–607.

Russell, S. J. & Norvig, P. (2016), *Artificial intelligence: a modern approach*, Pearson.

Samworth, R. J. (2012), 'Optimal weighted nearest neighbour classifiers'.

Sanders, G. T., Ott, N. M., Krystosek, P., Shimeall, T. & PA, C.-M. U. P. (2020), 'Network traffic analysis with silk: Analysts handbook for silk versions 3.15. 0 and later'.

Schmidt, B., Al-Fuqaha, A., Gupta, A. & Kountanis, D. (2017), 'Optimizing an artificial immune system algorithm in support of flow-based internet traffic classification', *Applied Soft Computing* **54**, 1–22.

sFlow (2024), 'sampled flow'. Accessed: 2024-07-07.
URL: *https://sflow.org/*

Shahbar, K. & Zincir-Heywood, A. N. (2017), 'Anon17: Network traffic dataset of anonymity services', *Faculty of Computer Science Dalhousie University, Tech. Rep* .

Shapira, T. & Shavitt, Y. (2019), Flowpic: Encrypted internet traffic classification is as easy as image recognition, *in* 'IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)', IEEE, pp. 680–687.

Shi, H., Li, H., Zhang, D., Cheng, C. & Wu, W. (2017), 'Efficient and robust feature extraction and selection for traffic classification', *Computer Networks* **119**, 1–16.

Sivanathan, A., Gharakheili, H. H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A. & Sivaraman, V. (2018), 'Classifying iot devices in smart environments using network traffic characteristics', *IEEE Transactions on Mobile Computing* **18**(8), 1745–1759.

Stallings, W. (2015), *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*, Addison-Wesley Professional.

Stockhammer, T. (2011), Dynamic adaptive streaming over http– standards and

design principles, *in* 'Proceedings of the second annual ACM conference on Multimedia systems', pp. 133–144.

Sun, G., Chen, T., Su, Y. & Li, C. (2018), 'Internet traffic classification based on incremental support vector machines', *Mobile Networks and Applications* **23**, 789–796.

Sun, G.-L., Xue, Y., Dong, Y., Wang, D. & Li, C. (2010), An novel hybrid method for effectively classifying encrypted traffic, *in* '2010 IEEE Global Telecommunications Conference GLOBECOM 2010', IEEE, pp. 1–5.

Sun, Y., Jiang, Y. & Tian, Y. (2010), 'Haar wavelets in data analysis', *Advanced Materials Research* **121-122**, 346 – 349.

Sun, Y., Valgenti, V. C. & Kim, M. S. (2011), Nfa-based pattern matching for deep packet inspection, *in* '2011 proceedings of 20th international conference on computer communications and networks (ICCCN)', IEEE, pp. 1–6.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1–9.

Tan, X., Fang, W. & Qu, Y. (2013), 'Network traffic prediction algorithm based on wavelet transform', *International Journal of Advancements in Computing Technology* **5**(5), 183–190.

Tanenbaum, A. S. & Wetherall, D. (2011), 'The network layer', *Computer Networks* pp. 343–423.

Thomas, A. J., Walters, S. D., Petridis, M., Malekshahi Gheytassi, S. & Morgan, R. E. (2016), Accelerated optimal topology search for two-hidden-layer feed-forward neural networks, *in* 'International Conference on Engineering Applications of Neural Networks', Springer, pp. 253–266.

Thomas, C., Sharma, V. & Balakrishnan, N. (2008), Usefulness of darpa dataset for intrusion detection system evaluation, *in* 'Data Mining, Intrusion detection, information assurance, and data networks security 2008', Vol. 6973, SPIE, pp. 164–171.

Tong, V., Tran, H. A., Souihi, S. & Mellouk, A. (2018), A novel quic traffic classifier based on convolutional neural networks, *in* '2018 IEEE Global Communications Conference (GLOBECOM)', IEEE, pp. 1–6.

Turing, A. M. (2009), *Computing machinery and intelligence*, Springer.

Ullah, S., Choi, J. & Oh, H. (2020), 'Ipsec for high speed network links: Performance analysis and enhancements', *Future Generation Computer Systems* **107**, 112–125.

Waked, L., Mannan, M. & Youssef, A. (2020), 'The sorry state of tls security in enterprise interception appliances', *Digital Threats: Research and Practice* **1**(2), 1–26.

Wang, P., Ye, F., Chen, X. & Qian, Y. (2018), 'Datanet: Deep learning based encrypted network traffic classification in sdn home gateway', *IEEE Access* **6**, 55380–55391.

Wang, W., Zhu, M., Wang, J., Zeng, X. & Yang, Z. (2017), End-to-end encrypted traffic classification with one-dimensional convolution neural networks, *in*

'2017 IEEE international conference on intelligence and security informatics (ISI)', IEEE, pp. 43–48.

Wang, W., Zhu, M., Zeng, X., Ye, X. & Sheng, Y. (2017), Malware traffic classification using convolutional neural network for representation learning, *in* '2017 International conference on information networking (ICOIN)', IEEE, pp. 712–717.

Wang, X. & Parish, D. J. (2010), Optimised multi-stage tcp traffic classifier based on packet size distributions, *in* '2010 Third International Conference on Communication Theory, Reliability, and Quality of Service', IEEE, pp. 98–103.

Wang, Z. (2015), 'The applications of deep learning on traffic identification', *BlackHat USA* **24**(11), 1–10.

Weinberger, K. Q. & Saul, L. K. (2009), 'Distance metric learning for large margin nearest neighbor classification.', *Journal of machine learning research* **10**(2).

White, G., Sundaresan, K. & Briscoe, B. (2019), 'Low latency docsis: Technology overview', *Research & Development* .

Williams, N., Zander, S., Armitage, G. et al. (2006), 'Evaluating machine learning algorithms for automated network application identification', *Center for Advanced Internet Architectures, CAIA, Technical Report B* **60410**, 2006.

Witten, I. H. & Frank, E. (2002), 'Data mining: practical machine learning tools and techniques with java implementations', *Acm Sigmod Record* **31**(1), 76–77.

Yan, H., Li, H., Xiao, M., Dai, R., Zheng, X., Zhao, X. & Li, F. (2019), Pgsm-dpi: Precisely guided signature matching of deep packet inspection for traffic

analysis, *in* '2019 IEEE Global Communications Conference (GLOBECOM)', IEEE, pp. 1–6.

Yang, L., Finamore, A., Jun, F. & Rossi, D. (2021), 'Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset', *IEEE Transactions on Network and Service Management* **18**(4), 4103–4118.

Ye, J., Li, J., Jiang, A. & Wu, C. (2021), Pf_ring-ta: A traffic-adaptive method of packet capturing by enhancing pf_ring, *in* '2021 IEEE 6th International Conference on Smart Cloud (SmartCloud)', IEEE, pp. 51–56.

Yuan, R., Li, Z., Guan, X. & Xu, L. (2010), 'An svm-based machine learning method for accurate internet traffic classification', *Information Systems Frontiers* **12**, 149–156.

yung (2016), 'Ustc-tfc2016 dataset'. Accessed: 2024-07-07.
URL: *https://github.com/yungshenglu/USTC-TFC2016/*

Zeiler, M. D. & Fergus, R. (2014), Visualizing and understanding convolutional networks, *in* 'Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13', Springer, pp. 818–833.

Zhang, H., Lu, G., Qassrawi, M. T., Zhang, Y. & Yu, X. (2012), 'Feature selection for optimizing traffic classification', *Computer Communications* **35**(12), 1457–1471.

Zhao, J., Jing, X., Yan, Z. & Pedrycz, W. (2021), 'Network traffic classification for data fusion: A survey', *Information Fusion* **72**, 22–47.

Zhou, H., Wang, Y., Lei, X. & Liu, Y. (2017), A method of improved cnn traffic classification, *in* '2017 13th international conference on computational intelligence and security (CIS)', IEEE, pp. 177–181.

Zou, Y., Zhu, J., Wang, X. & Hanzo, L. (2016), 'A survey on wireless security: Technical challenges, recent advances, and future trends', *Proceedings of the IEEE* **104**(9), 1727–1765.