CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Mobile Traffic Classification Over VPN

Evaluating Encrypted Traffic Classification Techniques on VPN Traffic: A Comparative Study

Master's thesis in Computer science and engineering

Nahusenay Yifter
Hassan Ghalayini

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Mobile Traffic Classification Over VPN

Evaluating Encrypted Traffic Classification Techniques on VPN Traffic: A Comparative Study

Nahusenay Yifter
Hassan Ghalayini

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Evaluating Encrypted Traffic Classification Techniques on VPN Traffic: A Comparative Study
Hassan Ghalayini Nahusenay Yifter

Supervisor: Romaric Duvignau, Computer science and engineering
Examiner: Risat Pathan, Computer science and engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

A Chalmers University of Technology Master's thesis template for LaTeX
Evaluating Encrypted Traffic Classification Techniques on VPN Traffic: A Comparative Study
NAHUSENAY YIFTER
HASSAN GHALAYINI
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In recent years, mobile network traffic classification has gained significant attention from network operators to better understand customer needs and allocate bandwidth based on application requirements. Research on machine learning and deep learning models has increased in popularity, as these methods enable more accurate classification while leveraging different aspects of network packets beyond just the payload.

The primary goal of this thesis is to compare the performance of different state-of-the-art deep learning models namely, Convolutional Neural Networks (CNNs), Recurrent Neural Networks(RNNs), and Autoencoders(AEs) through a series of experiments and to evaluate the feasibility of deploying these models for network classification for mobile traffic in VPN environments. The study focuses on network packets that are both encrypted and tunneled over Virtual Private Networks (VPN). A dataset of 50GB of VPN data is used to train, assess, and enhance analysis and training of the models. Our results indicate that CNNs effectively extract features but struggle with capturing sequential dependencies. By comparison, RNNs demonstrate greater efficiency in recognising temporal patterns and achieve higher recall rates. Autoencoders perform well for specific application classes but exhibit lower precision and recall overall.

This thesis suggests further investigation into a combined approach between convolutional neural networks and recurrent neural networks to be used for traffic classification over VPNs.

Keywords: Traffic Classification, VPN, CNN, RNN, Autoencoders, , Mobile Networks.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, Romaric Duvignau, for helping us along the way and providing support and resources on every meeting. We are also grateful to our examiner, Risat Pathan, for his valuable feedback on our midterm presentation and final presentation. Special thanks to Anthony Norman for the valuable writing seminars that helped us in writing our final report.

Nahusenay Yifter, Hassan Ghalayini, Gothenburg, 2025-10-07

Contents

List of Figures

List of Tables

1

Introduction

Traffic classification on mobile networks (TC) is a crucial task for network operators, as it can enable many applications and services that can benefit both the operators and the users. TC can help improve quality of service (QoS) for mobile users by providing network operators with traffic flows that can help them decide which traffic flow to prioritise or block based on their application types or service requirements [1]. For example, TC can help allocate more bandwidth to video streaming or online gaming applications that require high QOS, while limiting or blocking unwanted or malicious traffic such as spam or malware. TC can also be used to protect the network by providing results that aid network operators in monitoring and analysing mobile network traffic patterns of different applications and detecting any anomalous or malicious behaviour, such as Denial of Service attacks, botnets, or network intrusions. In addition, TC can provide support with forecasting the mobile network demands and trends by giving results that help to determine which applications and services are the most used by mobile users, and how they change over time and location. This can help network operators plan and optimise their network resources and infrastructure accordingly.

The aforementioned benefits and potentials of TC are particularly relevant for high-performance packet infrastructures such as the 5G mobile packet core, which is expected to support a variety of use cases and scenarios, such as enhanced mobile broadband, massive machine-type communications, and ultra-reliable and low-latency communications. However, TC faces a major challenge due to the increasing usage of Virtual Private Network (VPN) services by end-users, driven by privacy concerns and geoblocking issues. VPN services encrypt and tunnel the network traffic, making it harder for network operators to identify and classify the traffic. VPN services also hide important information that is usually used in TC, such as the port numbers, the destination IP address of the tunnelled packets, and the application signatures. This challenge has created the need for TC methods that are less affected by packet encryption and tunnelling of VPN, and that can rely on other features and techniques to classify the traffic.

Earlier research introduced TC using machine learning (ML), which is a branch of artificial intelligence that enables computers to learn from data and perform tasks that would otherwise require human intelligence. ML can offer several advantages for TC, such as the ability to handle large and complex data sets, to adapt to dynamic and evolving traffic patterns, and to discover new and unknown traffic classes. Several

state-of-the-art classification methods exist in the literature. Traditional machine learning approaches include Random Forest and k-Nearest Neighbours (kNN), and deep learning models such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs). Each ML classification method has different attributes, such as the input data (TCP payloads, HTTP fields, IP packets), the feature extraction and selection techniques (statistical, semantic, flow-based, etc.), the learning algorithm (supervised, unsupervised, semi-supervised, etc.), and the performance metrics (accuracy, precision, recall, etc.)[2].

The goal of this thesis is to compare several state-of-the-art ML classification methods to evaluate their performance in TC over VPN. We conduct a comprehensive experimental study using real-world VPN traffic data sets and compare the results of different ML classification methods in terms of various criteria, such as classification accuracy, computational complexity, scalability, robustness, and interpretability. Another goal of this thesis, rather than performance, is to analyse the potential privacy leakage of VPN when using ML classification methods. We investigate how much information can be inferred or revealed about the VPN users and their activities by applying ML classification methods to their encrypted and tunnelled traffic. We also discuss the ethical and legal implications of using ML for TC over VPN, and propose some possible solutions and recommendations to protect the privacy of VPN users.

1.1 Context

Over the past years, multiple revolutionary developments happened in the world of mobile telecom. Smartphones that contain mobile apps have been released. The apps have different categories such as social media, instant messaging, video sharing and streaming, or online gaming. The popularity of smartphones and mobile apps led to substantial growth in the number of mobile users, leading to a huge increase in mobile broadband network traffic. From the perspective of network operators, the traffic generated by the apps falls into distinct categories such as streaming, VoIP, social media, and file transfer, each with unique QoS requirements. For example, VoIP demands low latency, while streaming applications require high throughput. As a result, allocating uniform bandwidth across all traffic types is inefficient and can lead to performance degradation or resource waste. The need to separate each traffic by its category and allocate different service requirements for each type of traffic category led to the introduction of the concept of network slicing in 5G networks which is defined as slicing a physical network into several logical networks, each one can provide tailored services for a distinct application scenario. 5G network slices represented by logically isolated and self-contained networks are flexible enough and highly customizable to accommodate diverse business-driven use cases simultaneously over the same network infrastructure [3].

The aforementioned developments have emerged in the research for mobile traffic classification. The main challenge for TC is encrypted traffic, which led to research on TC to evolve through the following stages: port-based, payload-based, and flow-based [4]. Port based relies on the well-known ports used by the applications. Port

based method lost efficiency after the introduction of port randomisation. The payload-based or Deep Packet Inspection relies on matching the payload, which cannot work as efficiently on encrypted packets. Flow-based uses ML models and depends on statistics or time features. Flow based algorithms using ML algorithms had their own challenges, such as needing frequent updates and the need for experts in the domain to obtain handcrafted features. The challenges of using flow based algorithms using ML based models have emerged in the usage of deep learning (DL) models that has automatic extraction of the feature without the need for humans which made this approach the most desirable in TC [4], [5].

1.2 Goals and Challenges

1.2.1 Goals

The main goal of the project is to test DL models for TC over a VPN network. This objective can be further defined as follows:

1. The primary goal is to design and conduct experiments that compare the performance of state-of-the-art deep learning-based traffic classification (DL-TC) methods in networks where traffic is tunneled over VPNs. We aim to produce clear, analyzable results on the effectiveness of each DL-TC method, providing detailed statistics to identify which methods excel under specific criteria. This involves a thorough literature review, training of DL models, model tuning for VPN networks, and conducting experiments on simulated networks.

2. The secondary goal focuses on analyzing the experiment results to assess the feasibility of applying DL-TC methods over VPN traffic. We aim to identify the method that demonstrates superior overall performance, providing valuable metrics for future research. Additionally, we seek to explore the impact of DL-TC on the privacy of VPN networks, aiming to illuminate any potential privacy compromises.

1.2.2 Challenges

Several challenges are anticipated, including:

1. The potential need to modify existing DL methods to accommodate the unique characteristics of VPN traffic, such as using unsorted network data flows, poses a significant technical challenge.

2. Although there has been some progress in research on TC over encrypted networks, the specific area of TC over VPN has not been explored much [6]. This particular domain lacks comprehensive literature, which results in a scarcity of benchmark studies or established precedents that could guide new research.

3. Additionally, the project faces a challenge in the scarcity of labeled datasets specifically for mobile traffic, which is crucial for training accurate and efficient

ML models. The absence of such specialized datasets complicates the development of models that can effectively understand and classify the nuanced patterns of mobile network traffic, further complicating the task of traffic classification.

1.3 Research Questions

To guide the investigation into the performance of various ML methods for TC over VPN, this thesis aims to address the following research questions:

RQ How do different DL-based traffic classification methods perform when applied to VPN-encrypted traffic?

RQ-a: What are the accuracy, precision, recall, F1-score, training time, and inference time for each ML method (e.g., CNN, RNN, Autoencoder) when applied to VPN traffic?

RQ-b: How do these performance metrics compare to those achieved with non-VPN encrypted traffic?

1.4 Scope

In this section, we outline the specific boundaries and focus areas for our research on mobile traffic classification over VPN using machine learning methods. The scope of this thesis is defined by the following key points:

To design and conduct experiments comparing the performance of various state-of-the-art DL methods for traffic classification over VPN-encrypted traffic, such as CNNs, Recurrent Neural Networks (RNNs), and Autoencoders (AEs), chosen due to their proven effectiveness in encrypted traffic classification tasks [7], their architectural diversity (spatial and temporal modeling), and their relatively low computational complexity compared to more recent models such as Transformers [8]. The evaluation will be based on metrics including accuracy, precision, recall, F1-score, training time, and inference time. We will conduct a comprehensive experimental study using real-world VPN traffic datasets.

The scope of this thesis is limited in the following ways: Firstly, it will only test previously used methods for encrypted traffic classification without modifying the parameters or seeking to discover a better model that has not been previously explored. Secondly, it will exclusively use pre-recorded datasets of VPN traffic rather than generating new data or utilizing real-time data collection methods. This approach ensures the focus remains on evaluating existing ML techniques within the specified constraints.

1.5 Thesis Outline

This thesis has six chapters, each addressing a specific component of the study on encrypted traffic classification over VPN using deep learning models:

- **Chapter 2 – Background:** Introduces key concepts and technologies relevant to this study, including VPNs, mobile traffic classification techniques, and DL models such as CNNs, RNNs, and AEs. It also outlines the challenges associated with encrypted traffic and performance evaluation metrics. The chapter concludes with a review of related work on traffic classification in encrypted and VPN environments.
- **Chapter 3 – Methods:** Describes the methodology employed in this thesis, including data preparation, preprocessing steps, model architecture design, and training procedures for the CNN, RNN, and AE models. The chapter also outlines the experimental setup and evaluation metrics.
- **Chapter 4 – Evaluations:** Presents the experimental results obtained from the implemented models. This includes classification accuracy, precision, recall, F1-scores, confusion matrices, and training/inference time comparisons, along with a discussion of model performance.
- **Chapter 5 – Conclusion:** Summarises the key findings of the research, discusses their implications, and highlights limitations. The chapter concludes by proposing potential directions for future work in encrypted traffic classification.

2

Background

This study focuses on evaluating various ML techniques for encrypted traffic classification in VPN networks. The research explores DL approaches, including CNNs, RNNs and autoencoders, to enhance classification accuracy while addressing challenges posed by encryption and privacy-preserving protocols. By utilizing real-world VPN traffic datasets, the study aims to compare classification methods based on key performance metrics. This section provides the necessary background and theoretical foundation for understanding the studys methodology and findings, covering topics such as VPN traffic, DL, and comparative evaluation of classification models.

2.1 VPN

A VPN is a technology that creates a secure, encrypted connection over a less secure network, such as the internet. This secure connection, often referred to as a "tunnel", ensures that data transmitted between the user's device and the destination is protected from eavesdropping, censorship, and interference. VPNs are widely used to enhance privacy, secure data transmission, and access restricted resources [9]. Figure 2.1 below shows VPN operation, where user traffic is encrypted and tunnelled through a VPN server.

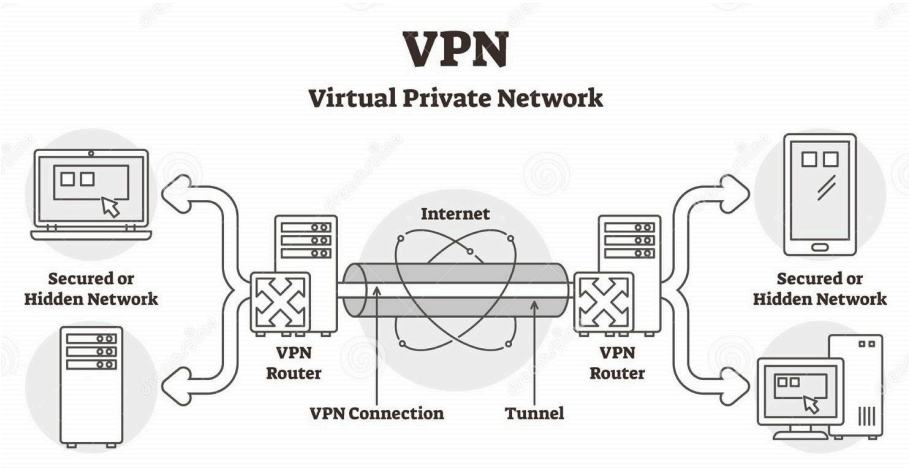

Figure 2.1: Diagram illustrating how a VPN works [10].

2.1.1 Types of VPNs

VPNs have different types to fulfill different applications and usecases. To know the different types of VPNs is essential to understand how they will provide secure connections.

- **Remote Access VPN:** Allows individual users to connect to a private network remotely. This is commonly used by employees to access their company's network securely from various locations [11].
- **Site-to-Site VPN:** Connects entire networks to each other, typically used to link branch offices with a central office over the internet. This setup ensures that resources from one network are available to users on the other network [11].
- **Client-Based VPN:** Installed on individual devices, this type of VPN secures the device's internet connection by routing it through the VPN server, ensuring encrypted communication.

2.1.2 Common VPN Protocols

The functionality of the VPN depends on the underlying protocols. Those protocols organizes the encryption, transmission, and tunnelling of data.

- **Internet Protocol Security (IPsec):** A suite of protocols designed to secure internet communications by authenticating and encrypting each IP packet in a data stream [12].
- **Layer 2 Tunneling Protocol (L2TP):** Often combined with IPsec for security, L2TP creates a tunnel between two L2TP connection points and is commonly used for site-to-site VPNs [13].
- **OpenVPN:** An open-source protocol that utilises SSL/TLS for encryption, offering a good balance between security and performance [14].
- **WireGuard:** A newer protocol known for its simplicity and high performance. It uses state-of-the-art cryptography and aims to be more efficient than older protocols [14].

2.1.3 Challenges in VPN Traffic Classification

As VPNs usage increase, mobile network traffic classification has became more complex and classifying VPN traffic presents several challenges [15]:

- **Encryption:** VPNs encrypt data, making it difficult for traditional traffic analysis tools to inspect the payload of packets.
- **Tunnelling :** By encapsulating data packets within another packet, VPNs hide the original source and destination information, making it challenging to determine the nature of the traffic.

- **Port Obfuscation:** Many VPNs use common ports (like 443 for HTTPS) to disguise their traffic, making it harder to distinguish VPN traffic from regular encrypted web traffic.

These challenges necessitate advanced methods, such as ML and DL techniques, to effectively classify and manage VPN traffic.

2.1.4 Applications of VPNs

- **Privacy Protection:** VPNs mask users' IP addresses and encrypt their online activities, safeguarding against tracking and surveillance.
- **Accessing Restricted Content:** Users can bypass geo-restrictions and censorship to access content that may be blocked in their region.
- **Secure Remote Work:** Organizations enable employees to securely access internal networks and resources from remote locations, ensuring data integrity and confidentiality.

2.2 Mobile networks traffic classification

Mobile network traffic classification refers to the process of identifying and categorising different types of data flows within a mobile network [16]. It involves analyzing network traffic patterns to determine the nature of the data being transmitted, such as whether it is web browsing, video streaming, voice calls, or other applications.

There are several benifits for mobile networks traffic classification such as:

- Efficiently managing network resources: By understanding the types of traffic, network operators can allocate resources effectively.
- Quality of Service: Different applications have varying QoS requirements. Proper classification ensures that critical services receive the necessary resources.
- Security: Detecting anomalous or malicious traffic patterns helps protect against cyber threats [7].

The earliest method for mobile network traffic classification relied on statistical features, particularly by analysing burst traffic patterns to identify application-specific behaviours [4]. Recently, the method of traffic classification has shifted from analysing statistical features into depending on Machine Learning, which can automatically classify traffic based on historical data and Deep Packet Inspection (DPI), which examines packet contents to identify applications and protocols [16].

There are several common categories for mobile network traffic such as Audio Streaming that includes services like music streaming platforms, Social Networking that includes traffic from social media apps, Video Calling data related to Video Calls, Video Streaming traffic from video streaming services, and traditional voice calls

over the network. In the context of 5G networks, traffic classification plays a crucial role in enabling efficient network slicing, where specific slices cater to different application requirements. By dynamically classifying traffic, we can ensure better resource utilisation and prioritise critical data with minimal packet loss or jitter.

2.3 Deep Learning

DL is a subset of machine learning that utilises neural networks with multiple layers to progressively extract meaningful representations from raw input data [17]. These algorithms are capable of learning hierarchical patterns, where lower layers detect fundamental features such as edges, while deeper layers capture more abstract concepts like shapes, objects, and even semantic meaning [18]. Deep Neural Networks (DNNs) are multi-layered artificial neural networks that form the foundation of modern DL techniques. Each layer consists of interconnected nodes (neurons) that transform the input data via weighted computations and non-linear activations [19]. DNNs have significantly advanced applications across various fields, including computer vision, natural language processing, bioinformatics, and climate science. In computer vision, they power object recognition, image segmentation, and medical imaging [20]. In natural language processing, they drive advancements in speech recognition, text generation, and machine translation [21]. They also contribute to bioinformatics, helping analyze biological data and predict protein structures [22], and support climate science by improving weather modelling and climate pattern prediction [23].

2.3.1 Convolutional Neural Networks

A CNN is a specialised deep learning architecture designed primarily for processing structured grid-like data, such as images [24]. CNNs are widely used in visual recognition tasks due to their ability to extract spatial features hierarchically [20].

CNNs utilise convolutional layers that apply small, learnable filters to the input to detect spatial features. Pooling layers reduce dimensionality and computational complexity while preserving essential information. Finally, fully connected layers integrate the extracted features to perform classification tasks.

The advantages of CNNs include translation invariance, where patterns are recognised regardless of their position in an image, and efficient feature extraction, reducing the number of parameters compared to traditional fully connected networks [17].

CNNs are widely used in medical imaging for disease detection, MRI and CT scan analysis [25], autonomous vehicles for object detection and scene segmentation [26], and security systems for facial recognition and surveillance [27]. Figure 2.2 shows a typical 1D CNN architecture, with convolutional and pooling layers before passing them to fully connected layers for classification.

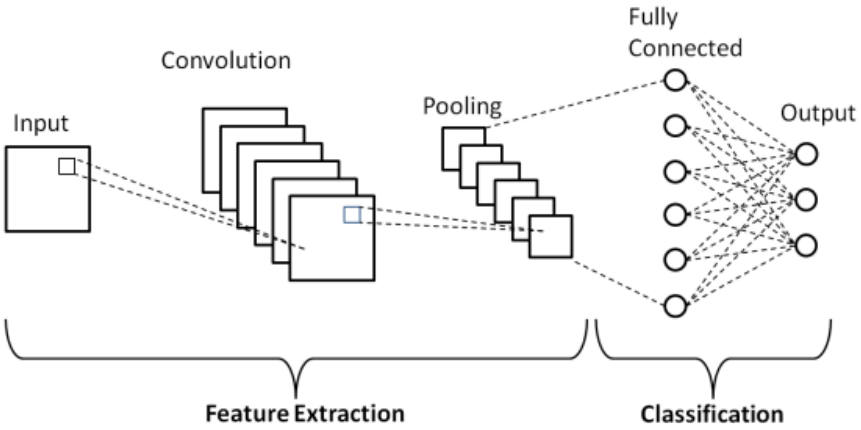

Figure 2.2: Diagram of a basic CNN architecture.

2.3.2 Recurrent Neural Networkss

RNNs are a class of neural networks designed for sequential data processing [28]. Unlike traditional Feed-Forward Networks (FFNs), RNNs incorporate memory by maintaining hidden states that carry contextual information across time steps. They process sequential data, making them ideal for tasks such as speech recognition, time-series forecasting, and natural language modelling (see Figure 2.3 for a comparison with feedforward networks).

Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) are specialised architectures that address the vanishing gradient problem and enhance long-range dependency modelling [28], [29].

Applications of RNNs include automatic transcription and voice assistants in speech recognition [30], financial forecasting for stock market trend prediction [31], and machine translation and text summarisation in language modelling [32].

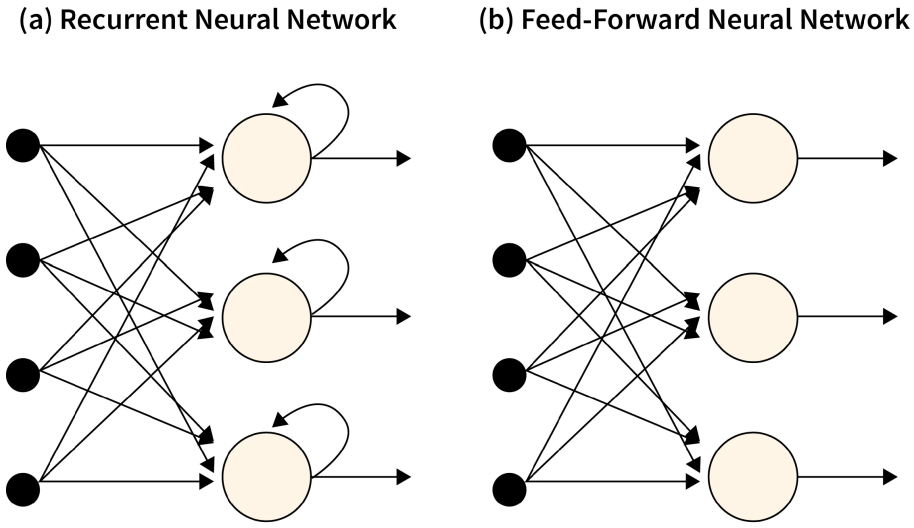

Figure 2.3: Comparison of RNN and Feedforward Neural Network architectures.

2.3.3 Autoencoders

AEs are a class of unsupervised neural networks used for feature extraction, dimensionality reduction, and anomaly detection [33]. They consist of an encoder that compresses input data into a lower-dimensional latent representation and a decoder that reconstructs the original data from this representation, as illustrated in Figure 2.4.

Various types of autoencoders exist, including denoising autoencoders, which remove noise from corrupted inputs [34]; variational autoencoders (VAEs), which generate new data samples by learning probabilistic distributions [35]; and sparse autoencoders, which encourage sparsity in the latent representation for better feature selection [36].

Autoencoders are widely used in anomaly detection, such as fraud detection and cybersecurity threat analysis [37], data compression for efficient encoding of high-dimensional data [33], and image generation using VAEs [35].

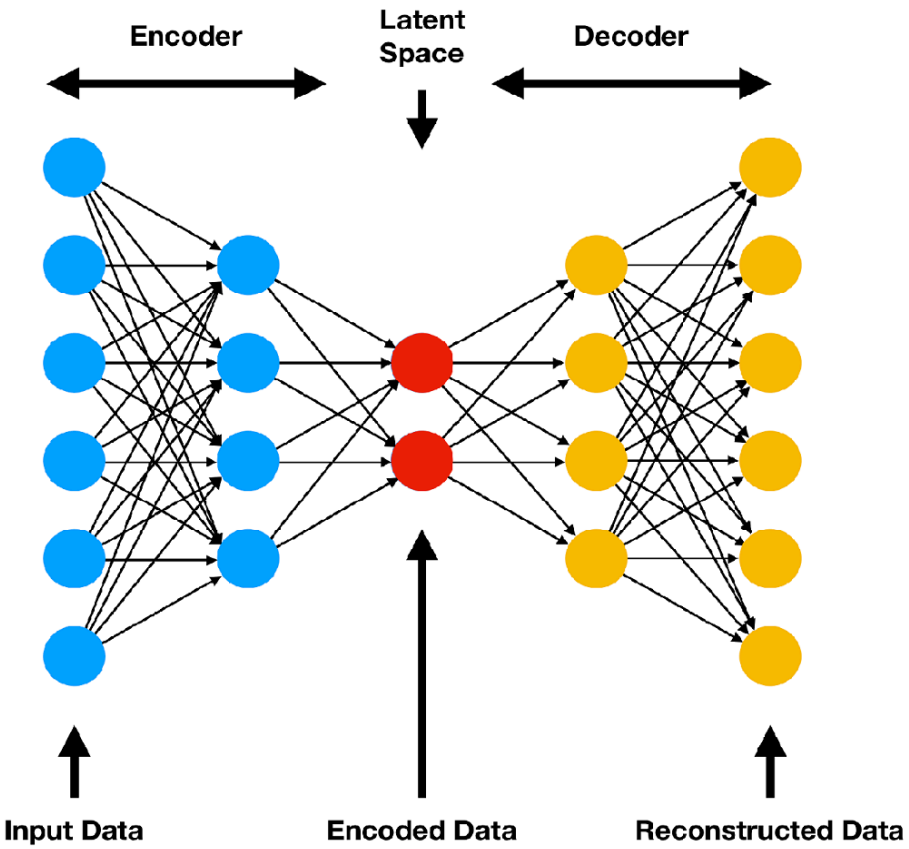

Figure 2.4: Diagram of a basic Autoencoder architecture.

2.3.4 Performance Metrics

Performance metrics are fundamental for evaluating the effectiveness and reliability of ML models, particularly in the context of encrypted TC. These metrics provide

quantitative measures that enable an in-depth assessment of the model's performance in terms of classification accuracy, precision, recall, loss, and computational efficiency.

2.3.4.1 Accuracy

Accuracy measures the proportion of correctly classified samples among the total number of samples. While it provides a general evaluation of model performance, it may not be a sufficient metric when dealing with imbalanced datasets. Mathematically, accuracy is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.1}$$

where TP, TN, FP, and FN denote True Positives, True Negatives, False Positives, and False Negatives, respectively.

2.3.4.2 Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's predictions compared to the true labels. It visualizes the distribution of correct and incorrect predictions across all classes, enabling the identification of patterns in misclassifications. Table 2.1 below shows the structure of a confusion matrix.

Table 2.1: Structure of a confusion matrix.

	Predicted Positive	**Predicted Negative**	
Actual Positive	True Positive (TP)	False Negative (FN)	
Actual Negative	False Positive (FP)	True Negative (TN)	

2.3.4.3 Precision

Precision quantifies the proportion of correctly predicted positive samples out of all predicted positive samples. This metric is particularly important in scenarios where false positives have significant implications. Precision is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.2}$$

2.3.4.4 Recall (Sensitivity)

Recall, also known as sensitivity, measures the proportion of actual positive samples that are correctly identified by the model. This metric is especially critical in applications where missing positive instances (false negatives) carries a high cost. Recall is calculated as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.3}$$

2.3.4.5 F1-Score

The F1-Score is the harmonic mean of precision and recall, offering a balanced evaluation when the dataset is imbalanced. It is defined as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

2.3.4.6 Loss

Loss functions play a critical role in guiding the training process of machine learning models by quantifying the error between predicted and actual values. Minimizing the loss ensures that the model learns patterns effectively from the training data. Common types of loss functions include:

- **Mean Squared Error (MSE):** Used for regression tasks, MSE calculates the average squared difference between predicted and actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

- **Cross-Entropy Loss:** Commonly used for classification tasks, this loss measures the difference between the predicted probability distribution and the true distribution.

$$\text{Cross-Entropy Loss} = -\sum_{i} y_i \log(\hat{y}_i) \tag{2.6}$$

- **Hinge Loss:** Often used for binary classification with Support Vector Machines, hinge loss ensures a margin between the classes.

The choice of loss function depends on the task and the specific requirements of the model being trained.

2.3.4.7 Computational Efficiency

Computational efficiency metrics assess the practicality of the model in real-world scenarios:

- **Training Time:** The time required for the model to complete the training process.
- **Inference Time:** The time taken by the model to generate predictions on new data.

These metrics are crucial in applications where real-time or near-real-time predictions are essential.

2.4 Related Works

Over the recent years, the usage of VPNs has increased. VPNs allows the user to establish a private connection that protects the data using tunnelling and encryption. VPNs make traditional TC using port-based or payload based ineffective since VPNs hide important information such as IP address, port, or protocol [4]. VPNs also use encryption which obscures the payload of the packets as well. Flow based TC using ML and DL have their own challenges over VPNs as well such as diversity, variability, and unpredictability of mobile network traffic making it difficult to generate traffic patterns and features dynamically and heterogeneously [38]. There are existing researches in the area that discuss using ML based classification methods over encrypted traffic.

Aceto et al (2019) study TC of a network with encrypted traffic [7]. It compares several ML-based classification methods and architectures providing a framework for the comparison and performance evaluation of accuracy, speed, resource consumption, privacy, and adaptability. The paper concludes that DL is a promising and viable strategy for mobile encrypted traffic classification, but also highlights the need for further research and development to address the limitations and challenges of the current DL models. One of the main highlighted challenges discussed in the paper is the high requirement for training data. An equally important challenge mentioned is the quality and purity of the samples used for training. Another challenging aspect mentioned in the paper is that DL models have several parameters to be tuned and searching those parameters might be as challenging as manually extracting features in the traditional ML models. An important takeout from the paper is that **there is no DL classification method that could be described as the best method**, and the performance of each DL method is heavily related to the nature of the input data used in that model, the more it is relevant the better the results of the DL method.

Wang et al [4] propose a general framework of DL based mobile services encrypted traffic classification and review the most recent existing work according to classification task definition, data preparation, pre-processing, model input design, pre-training design and model architecture. The paper also highlights a noteworthy issue of deep learning in encrypted traffic classification. The first issue mentioned in the paper was that some datasets used by the existing work were imbalanced because no dataset contains all application traffic and it is very hard and time-consuming to cover all the network scenarios in the datasets. The existing methods to overcome the imbalance are modifying the objective cost function, under-sampling, and over-sampling in addition to generating artificial datasets. A promising existing research direction for TC mentioned in this paper is the concept of **semi-supervised learning** based traffic classification, as the supervised way need big amount of labeled data that has high cost and high difficulty to find. The aforementioned proposed semi-supervised way consists of two stages of training. The first stage is for training a large amount of unlabeled data then the next stage will have the training of only a small amount of labeled data.

Afuwape et al [39] compare individual and collective performances with state-of-the-

art Ensemble and Single models that show which algorithms are better than others in both VPN and non-VPN networks by developing traffic classification algorithms performed on supervised machine learning. The result of their experiment showed that ensemble algorithms had better performance than single machine learning classifiers. They achieved the highest accuracy with the Random Forest classifier using non-VPN traffic and VPN traffic. For their future work, Security models will be investigated such as the effect of Denial of Service attacks (DoS).

Lin et al. (2022) [8] propose ET-BERT, a novel pre-training model for encrypted traffic classification, leveraging large-scale unlabeled traffic to learn robust datagram-level representations. The ET-BERT model incorporates two pre-training tasks: Masked BURST Model (MBM) and Same-origin BURST Prediction (SBP). These tasks capture contextual relationships within traffic, enabling superior performance across multiple encrypted traffic classification tasks. The study achieves state-of-the-art results, particularly in scenarios like VPN and TLS 1.3 traffic classification, highlighting the potential of pre-trained Transformer-based architectures in addressing the challenges of imbalanced and diverse encrypted datasets.

Building on this direction, Yuxiang et al. (2024) [40] proposed a balanced supervised contrastive learning approach for encrypted traffic classification, employing a hybrid encoder with self-attention transformers and GLSTM to capture both global and temporal dependencies. The method applies a class-balanced contrastive loss to address class imbalance and utilizes a two-stage training scheme with contrastive pre-training followed by supervised fine-tuning, demonstrating improved performance on imbalanced datasets.

Using a traditional ML pipeline, Saber et al. (2018) [41] proposed a method that combines oversampling and undersampling to address class imbalance in encrypted traffic classification. After balancing the dataset, they applied PCA to reduce the feature dimensionality before training an Support Vector Machine (SVM) classifier on VPN and non-VPN traffic. This approach successfully reduced the features from 23 to 15 principal components and achieved high accuracy under different flow timeout settings.

Shapira and Shavitt (2021) [42] introduced FlowPic, a generic representation for encrypted traffic classification that converts flow-level packet sizes and arrival times into 2D histogram images. These images are fed into a CNN to classify traffic by category (e.g., browsing, chat, video, VoIP, file transfer) and even identify specific applications. Their method maintains high accuracy over 96% for category and up to 99.2% for VPN traffic, and above 89% for Tor traffic even when trained only on nonVPN data

Balachandran and Amritha (2022) [43] proposed a method to classify VPN network traffic by combining entropy-based features with time-related characteristics. They estimated entropy alongside timing features, then tested Random Forest, KNN, and Artificial Neural Network (ANN) classifiers to differentiate VPN from non-VPN traffic and identify applications. Their combined approach consistently achieved above 90% accuracy across all models

3

Methods

This section describes the methodology used for implementing machine learning models for traffic classification over VPN-encrypted traffic. The primary focus is on utilizing a CNN, RNN, and AE to classify VPN traffic data.

3.1 Data Preparation

The dataset used in this study consists of pre-recorded VPN traffic data, structured and processed to fit the requirements of our machine learning models.

3.1.1 Data collection

To prepare the datasets for deep learning methods, we initially used a locally collected dataset titled *5G mobile app traffic tracks - Chalmers 2023*, publicly available via IEEE Dataport [44]. This dataset consists of 1,912 pcap files, systematically organised into eight folders, each representing one of the following widely used mobile applications: Facebook, Instagram, LinkedIn, Spotify, TikTok, Twitter, Wikipedia, and YouTube. Each pcap file captures one minute of encrypted network traffic from a single application. The traffic was generated using a 5G-connected mobile device and routed through a VPN tunnel to anonymise content-level details. This ensured that only metadata, specifically packet size, direction, and timing, was available for analysis, aligning with privacy preserving standards in traffic classification research. The traces were collected over Chalmers University's 5G infrastructure during 2023, providing realistic mobile usage patterns under contemporary network conditions. This dataset is especially valuable for studying encrypted traffic in mobile environments, focusing on temporal and structural characteristics rather than payload-based features.

The VPN-nonVPN dataset provided by the Lincoln Laboratory of the Massachusetts Institute of Technology [45] is included in this study for cross-dataset evaluation. It contains encrypted traffic traces categorised by application types such as Skype, YouTube, and Netflix. The dataset does not distinguish whether the traffic originates from mobile or desktop devices; instead, it emphasises traffic typesuch as streaming, VoIP, chat, command and control, and file transferand network environment (VPN or non-VPN). This dataset is used to assess the generalization capability of the models under varying traffic behaviours and encryption contexts.

In addition to the datasets previously mentioned, the UNB ISCX [46] Network Traffic Dataset is utilised to enhance our analysis and model training. This dataset offers a comprehensive view of network traffic types, including web browsing activities through Firefox and Chrome, email exchanges via SMTPS, POP3S, and IMAPS, and various chat services such as ICQ, AIM, Skype, Facebook, and Hangouts. It also comprises streaming content from Vimeo and YouTube, file transfers through Skype, FTPS, and SFTP using Filezilla and external services, VoIP calls made via Facebook, Skype, and Hangouts, and P2P file sharing through uTorrent and Transmission. This diverse range of content and traffic types provides a rich foundation for training DL models on a variety of network behaviours and patterns, complementing the datasets from Chalmers and the Lincoln Laboratory of MIT.

3.1.2 Data Pre-Processing

Several approaches have been proposed for preprocessing encrypted traffic data, including raw-packet transformations [16], time-series feature extraction [7], image-based representations [4], and BERT-inspired tokenisation strategies [8]. This section describes two distinct pre-processing approaches: one used in Wang et al.'s [4] method and the other employed by ET-BERT [8]. The first approach converts raw packet data into image formats suitable for DL algorithms, while the second leverages structured data extraction techniques, which tokenise datagram-level fields into sequences for contextual learning in encrypted TC.

3.1.2.1 Wang et al. Data Pre-Processing

The data pre-processing approach in Wang et al. [4] involves two main steps: transforming raw packet traces into PNG images and subsequently converting these images into MNIST format for compatibility with DL techniques.

Raw Packet Trace to PNG

A script is employed to convert raw session data into PNG images, leveraging the capability of CNNs to process image data efficiently. The raw session data, consisting of packet captures, is structured into a two-dimensional numpy array where network traffic features (e.g., packet lengths, inter-arrival times) are mapped to pixel intensities. The array is normalised to a 0-255 range and then converted into a grayscale image using the Python Imaging Library (PIL).

To maintain consistency, normalisation is employed to adjust the data to a common scale, ensuring that varying feature ranges do not distort the image representation. Consequently, a grayscale image is created from the normalised values. The grayscale conversion preserves data integrity while enabling CNNs to extract spatial features effectively.

PNG to MNIST

This step converts the PNG images into a format compatible with the MNIST dataset structure, commonly used in image classifications with DL. This ensures compatibility with widely adopted DL frameworks without requiring architectural

modifications, as the MNIST format is well-optimised with fixed-size and grid-like inputs.

The script reads and ensures that the PNG images are in grayscale before resizing them to 28x28 pixels. The pixel values are normalised to the range [0,1] to enhance model training. The processed images and corresponding labels are stored in a compressed numpy file, mimicking the MNIST datasets structure.

3.1.2.2 ET-BERT Data Pre-Processing

ET-BERT [8] uses a different data preprocessing strategy tailored for encrypted traffic classification. Instead of image transformation, it extracts structured information from packet traces and represents it in a format suitable for DL models.

Packet Feature Extraction

The raw network traffic is parsed to extract meaningful features from each packet. These features include packet length, inter-arrival time, protocol type, payload entropy, and sequence order within a session. These extracted features serve as fundamental indicators of traffic behaviour.

Burst-Level Aggregation

Packets within the same session are grouped into bursts, capturing traffic flow dynamics. A burst is defined as a set of consecutive packets exchanged between a source and a destination within a short time window. This aggregation retains structural patterns in encrypted communication.

Feature Vector Construction

Each burst is converted into a structured feature vector by concatenating statistical summaries of its constituent packets. These include min/max/mean packet lengths, variance in inter-arrival times, protocol distributions, and entropy measures. The feature vectors serve as the input representation for encrypted traffic classification models.

This structured representation enables ET-BERT to effectively capture contextual patterns within encrypted traffic, offering an advanced approach compared to traditional feature extraction methods.

3.2 Model Architecture

This section outlines the specific architectures, including their layer composition and training settings, used for the DL models evaluated in this study.

3.2.1 Convolutional Neural Network

CNNs are particularly effective for capturing spatial hierarchies and local patterns within data. For this study, a 1D-CNN architecture was implemented to process

encrypted traffic data represented as a time series of packet sizes, inter-arrival times, and directions.

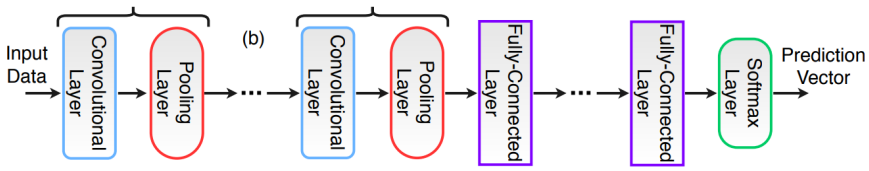

Figure 3.1: Illustration of the CNN architecture used in this study. Adapted from Aceto et al. (2019) [7].

The model shown in Figure 3.1 includes convolutional layers for feature extraction, pooling layers for dimensionality reduction, fully connected layers for feature integration, and a softmax layer for classification.

The CNN architecture employed in this study consists of the following layers and components:

Input Layer: The input layer accepts encrypted traffic data that has been preprocessed into a numerical format suitable for convolutional operations. Specifically, the raw packet capture data is converted into a fixed-length vector, where each vector represents features such as packet size, inter-arrival time, and byte distributions. These features are normalized to ensure uniform scaling and to enhance model performance. By transforming the data into this structured format, the model is better equipped to extract spatial dependencies inherent in encrypted traffic patterns.

Convolutional Layers: The first convolutional layer applies 32 filters of size 1x25, followed by a rectified linear unit (ReLU) activation function to introduce non-linearity. A max-pooling operation with a pool size of 2 is used to downsample the feature maps and reduce dimensionality. The second convolutional layer applies 64 filters of size 1x25, also followed by a ReLU activation function and max-pooling operation. These layers are designed to progressively capture local and abstract features from the input data, improving the model's ability to classify encrypted traffic patterns. Additional padding is applied to ensure that the spatial dimensions are preserved across layers, facilitating deeper analysis.

Fully Connected Layer: After flattening the output from the convolutional layers, a fully connected layer with 1024 neurons is applied. This layer aggregates the spatial features extracted by the convolutional layers and serves as a high-level feature integrator. To enhance the feature representation, batch normalization is applied before the activation function to stabilize training and accelerate convergence.

Dropout Layer: A dropout layer with a rate of 0.5 is included after the fully connected layer to prevent overfitting. During training, it randomly sets a fraction of the input units to zero, ensuring the model generalizes well to unseen data. This

technique enhances the robustness of the model against overfitting, especially given the high dimensionality of the fully connected layer.

Output Layer: The output layer employs a softmax activation function to produce a probability distribution over the class labels. Each neuron in the output layer corresponds to a traffic type, and the output values indicate the likelihood of each traffic type being present in the input data. This probabilistic interpretation allows for straightforward classification and enables effective performance evaluation using metrics such as accuracy, precision, and recall.

3.2.2 Recurrent Neural Networks Model Architecture

RNNs are widely used for sequential data modelling, making them particularly suitable for analysing network traffic flows. This study employs an RNN-based architecture, specifically utilising LSTM networks to classify encrypted network traffic effectively [7]. LSTMs are capable of learning long-term dependencies by maintaining internal memory states, making them ideal for processing time-series data such as network traffic. The model is designed to learn temporal dependencies within network packet sequences, distinguishing different traffic classes based on their sequential patterns.

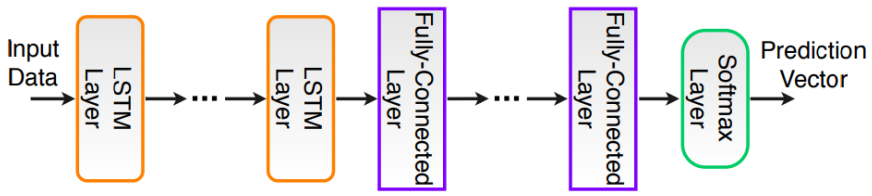

Figure 3.2: RNN model architecture used in this study.

The RNN model depicted in Figure 3.2 consists of multiple LSTM layers followed by fully connected layers and a final Softmax layer, which are detailed as follows:

- **LSTM Layer:** The core of the model is an LSTM layer, which extracts sequential patterns from network packet sequences. In this study, the LSTM layer consists of 50 memory units, which have been empirically chosen to balance computational efficiency and classification accuracy.
- **Dropout Layer:** To prevent overfitting, a dropout layer is applied after the LSTM layer. This layer randomly deactivates a fraction of the neurons during training, reducing reliance on specific features and enhancing model generalisation.
- **Dense (Fully Connected) Layer:** The final layer is a dense (fully connected) layer with a softmax activation function. This layer outputs probabilities for each traffic class, enabling multi-class classification of network flows.

Training Methodology: The model undergoes a structured training process to ensure optimal performance:

- **Epochs:** The model is trained for 20 full epochs, meaning the entire dataset is processed 20 times to refine the learned patterns.
- **Batch Size:** A batch size of 64 is used, meaning 64 samples are processed in each training iteration before updating model weights. This approach balances computational efficiency with learning stability.
- **Validation Data:** A portion of the dataset is reserved for validation, allowing the models performance to be evaluated on unseen data. This ensures that the model does not overfit to the training data and generalises well to real-world traffic scenarios.

3.2.3 Autoencoder

Autoencoders are unsupervised neural networks designed to learn efficient representations of input data by reconstructing it through a bottleneck structure. For this study, an autoencoder was employed to extract compressed features from VPN encrypted traffic data and identify patterns indicative of different traffic types.

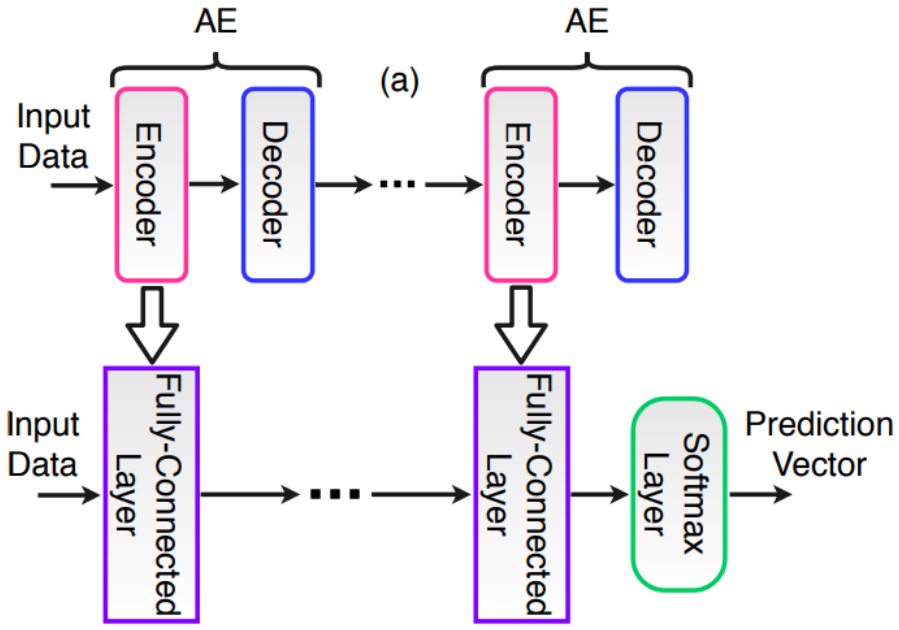

Figure 3.3: Illustration of the Autoencoder architecture used in this study.

The architecture in Figure 3.3 includes an encoder to compress input data into a latent representation and a decoder to reconstruct the original input [7].

Input Layer The input layer accepts normalised encrypted traffic data, represented as a 784-dimensional vector for each traffic instance. The data normalisation

process ensures that all input features are scaled between 0 and 1, improving model stability and convergence.

Encoder The encoder consists of three fully connected layers that progressively reduce the dimensionality of the input data:

- The first layer reduces the input dimension from 784 to 512 neurons and uses the ReLU activation function to introduce non-linearity.
- The second layer reduces the dimension further to 256 neurons, also using ReLU activation.
- The bottleneck layer compresses the data into a 128-dimensional latent space representation, capturing the most salient features of the input data.

Decoder The decoder mirrors the encoder's structure, reconstructing the input data from the compressed latent representation:

- The first layer expands the 128-dimensional latent space to 256 neurons, using ReLU activation.
- The second layer further expands the data to 512 neurons, also using ReLU activation.
- The output layer reconstructs the original 784-dimensional input, applying a sigmoid activation function to ensure that output values remain in the range of [0, 1].

Training and Optimization: The autoencoder was compiled using the Adam optimiser with a learning rate of 0.001 and trained to minimise MSE between the input and reconstructed data. Early stopping was employed to monitor validation loss and prevent overfitting. The model was trained for up to 50 epochs with a batch size of 32, and the best model weights were restored based on validation performance.

Loss Visualization: The training process was monitored by plotting the loss curves for both training and validation datasets. This visualisation provides insights into the convergence behaviour and generalisation of the autoencoder.

Output: The trained autoencoder produces a latent representation for each input instance, which can be further utilised for classification or anomaly detection tasks. The reconstruction loss can also serve as an indicator of anomalous traffic behaviour.

3.3 Evaluation Setup

This section describes the experimental setup used to evaluate the performance of the deep learning models applied to VPN-encrypted traffic classification.

Preprocessing All traffic data was preprocessed using the ET-BERT framework[8], which extracts temporal and statistical features from encrypted traffic flows. This method transforms pcap files into feature vectors. The extracted features were fed into classification models.

Model Architectures The models used in this study are adapted from Aceto et al. 2019 [7], which includes lightweight 1D CNNs and deep autoencoders designed for mobile traffic classification. The autoencoder is used both for reconstruction loss evaluation and as a feature extractor in a downstream Multi-Layer Perceptron(MLP) classifier.

Evaluation Metrics The models were evaluated using the following metrics:

- **Accuracy:** overall proportion of correctly classified instances.
- **Precision:** correctness of predicted positive samples.
- **Recall:** ability to identify actual positive samples.
- **F1-score:** harmonic mean of precision and recall.
- **Training Time and Inference Time:** computational efficiency.

Evaluation Settings All experiments were performed on a workstation with the following specifications:

- Intel Core i7-12700K CPU
- 8GB RAM
- NVIDIA RTX 1050Ti GPU
- Python 3.10 with TensorFlow and Keras

Experiment Design The evaluation was structured into the following experiments:

1. Train CNN, RNN, and AE on the Chalmers dataset and evaluate classification performance.
2. Use the same trained models (no retraining) to test generalization on MIT and UNB datasets.
3. Compare performance across datasets and analyze impact of class imbalance.
4. Evaluate computational costs (training/inference time).

This systematic design allows us to assess each model's performance both in-domain and out-of-domain, and to analyze trade-offs between accuracy and efficiency.

4

Evaluations

This section presents the outcomes of the model training and evaluation process, including key performance metrics such as training accuracy, test accuracy, precision, and recall. It will include a detailed analysis of the confusion matrix to illustrate class-wise performance and highlight patterns of misclassification. Additionally, it provides an in-depth discussion of how different models performed under different datasets, the challenges encountered, and the broader implications of the results.

4.1 Convolutional Neural Network

The CNN model is trained and evaluated using the Chalmers dataset [44], consisting of VPN encrypted traffic data. The results reveal key insights into the model's performance across eight traffic classes. Below is an interpretation of the confusion matrix and classification metrics:

Table 4.1: Classification Report for CNN over Encrypted Traffic

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Facebook	0.29	0.33	0.30	86
Instagram	0.53	0.28	0.37	99
Linkedin	0.45	0.76	0.57	80
Spotify	0.80	0.68	0.74	95
Tiktok	0.86	0.29	0.44	85
Twitter	0.41	0.56	0.47	89
Wikipedia	0.47	0.49	0.48	96
Youtube	0.68	0.78	0.73	82
Accuracy	0.52 (712 instances)			
Macro Avg	0.56	0.52	0.51	712
Weighted Avg	0.56	0.52	0.51	712

Table 4.1 summarizes the classification performance across eight classes for VPN-encrypted traffic. The reported metrics, including precision, recall, and F1-score, vary significantly across classes, reflecting the inherent challenges of classifying VPN-encrypted traffic.

Precision: The model achieves high precision for Tiktok (0.86) and Spotify (0.80), indicating its ability to correctly identify these traffic types with minimal false positives. However, precision for Facebook (0.29) and Instagram (0.53) suggests higher rates of false positives for these classes.

Recall: Linkedin exhibits the highest recall (0.76), indicating the model's strong ability to identify most instances of this class. Conversely, Tiktok (0.29) and Instagram (0.28) have lower recall, revealing difficulties in detecting these traffic types.

F1-Score: The overall F1-scores range from 0.30 (Facebook) to 0.74 (Spotify), reflecting the model's balanced performance for certain classes while struggling with others.

Accuracy: An overall classification accuracy of 52% highlights the complexities introduced by encryption in VPN traffic, which obfuscates many features typically used for classification.

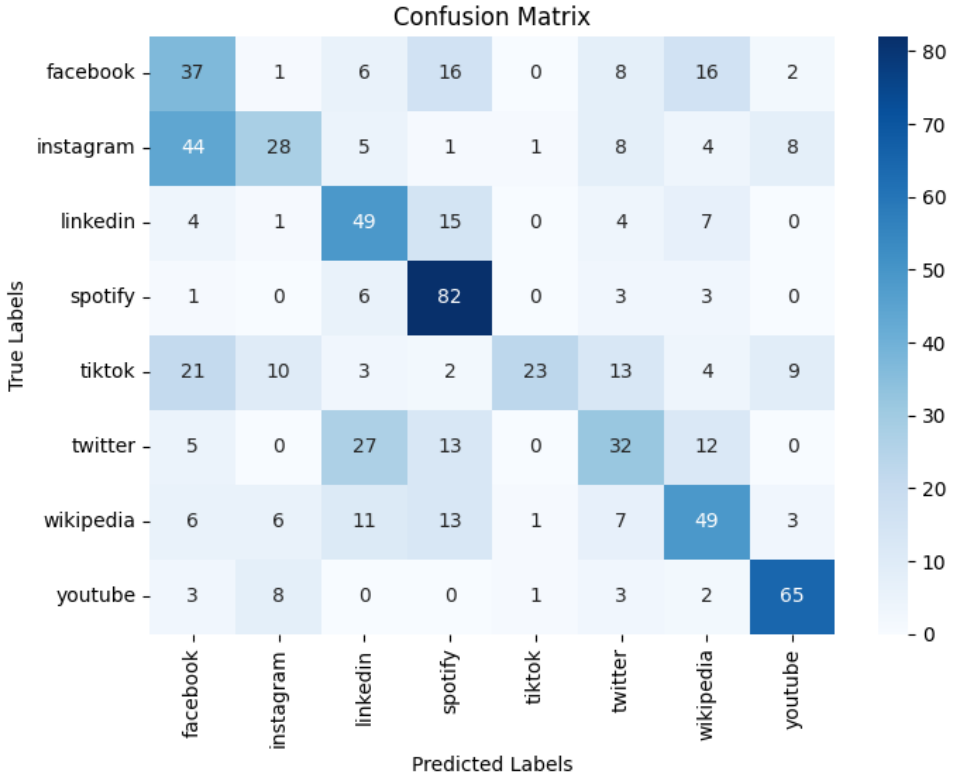

Figure 4.1: Confusion matrix illustrating the performance of the CNN model on the Chalmers dataset. The matrix highlights true labels versus predicted labels across eight traffic classes.

The confusion matrix, as seen in Figure 4.1, reveals that the CNN model performs well on certain applications, such as YouTube and Spotify, with relatively low misclassification. These applications typically generate continuous and high-volume

traffic flows with consistent burst patterns, which likely align well with the local feature extraction capabilities of convolutional layers. In contrast, the model struggles with applications like TikTok and Instagram, where misclassifications are more frequent. These platforms are characterised by dynamic, user-driven interactions such as frequent scrolling, switching between media types, and variable session lengths. These circumstances may produce less regular traffic patterns. As CNNs are inherently limited in capturing long-term temporal dependencies, they may fail to adequately represent the irregular bursts of such traffic. Additionally, confusion between similar app categories (e.g., Instagram and Facebook) suggests that traffic generated by social media platforms shares overlapping statistical characteristics, making precise classification more challenging.

4.2 RNN

The RNN model is trained and evaluated using the Chalmers dataset, similar to the CNN. Table 4.2 shows the result of the confusion matrix when using the RNN model.

Table 4.2: Classification Report for RNN over Encrypted Traffic

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Facebook	0.37	0.45	0.40	86
Instagram	0.60	0.42	0.49	99
Linkedin	0.54	0.80	0.65	80
Spotify	0.76	0.72	0.74	95
Tiktok	0.74	0.52	0.61	85
Twitter	0.53	0.63	0.57	89
Wikipedia	0.50	0.52	0.51	96
Youtube	0.72	0.80	0.76	82
Accuracy	0.58 (712 instances)			
Macro Avg	0.57	0.58	0.58	712
Weighted Avg	0.57	0.58	0.58	712

Table 4.2 summarizes the classification performance across eight classes for VPN-encrypted traffic using LSTM. Similar to CNN the reported metrics vary significantly and compared to CNN there is an improvement in recall that reduces false negatives.

Precision: The model achieves the highest precision for Spotify (0.76) and Youtube (0.72), indicating strong confidence in identifying these traffic types with relatively few false positives. However, precision for Facebook (0.37) and Twitter (0.53) remains lower, suggesting more false positives in these categories.

Recall: Linkedin (0.80) and Youtube (0.80) exhibit the highest recall, meaning the model effectively identifies most instances of these classes. In contrast, Tiktok (0.52) and Instagram (0.42) have lower recall, indicating challenges in correctly detecting all occurrences of these traffic types.

F1-Score: The overall F1-scores range from 0.40 (Facebook) to 0.76 (Youtube), demonstrating that the model performs well in certain categories but struggles with others, particularly Facebook and Instagram, which have relatively low F1-scores due to imbalanced precision and recall.

Accuracy: The RNN achieves 58% accuracy, outperforming the CNN by 6 percentage points. This relative improvement, along with higher recall on certain traffic classes, suggests that the model was better able to capture temporal dependencies, despite the overall task complexity. However, certain applications (e.g., Facebook and Instagram) still present classification challenges due to similar traffic patterns.

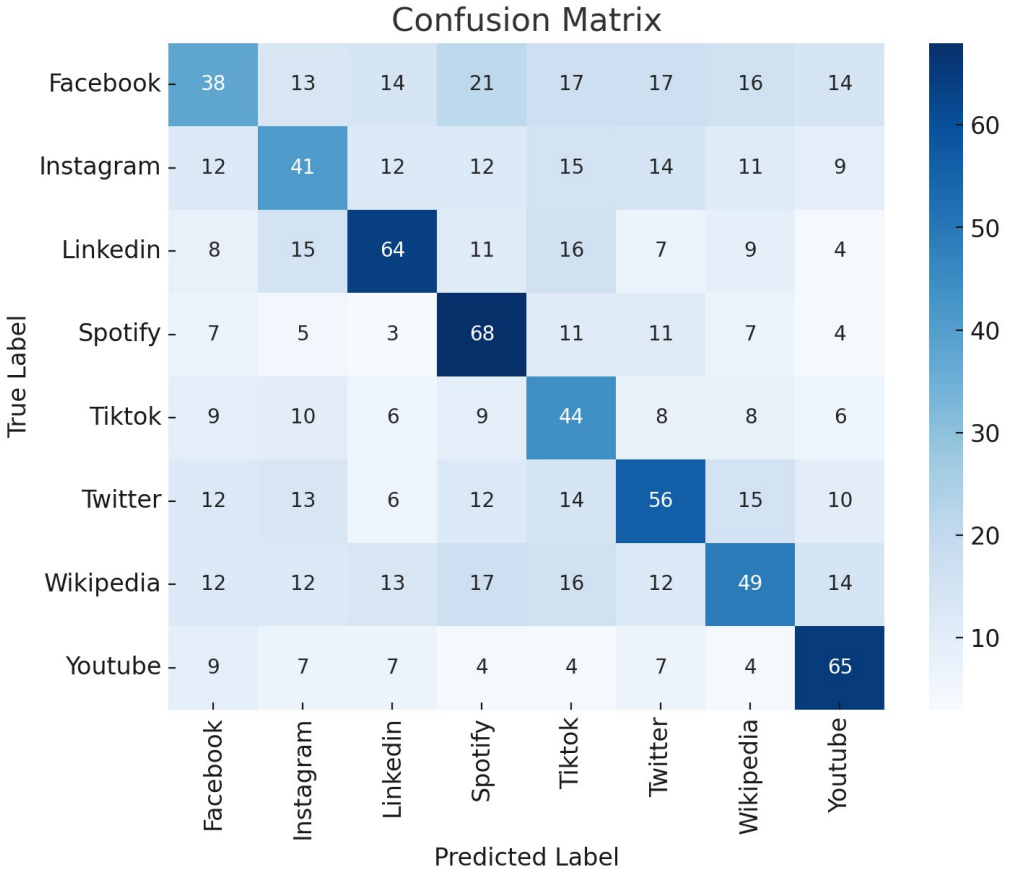

Figure 4.2: Confusion matrix illustrating the performance of the RNN model on the Chalmers dataset. The matrix highlights true labels versus predicted labels across eight traffic classes.

The RNN model as observed from Figure 4.2 showed notable improvements over the CNN in classifying traffic from interactive applications such as TikTok and Instagram. These platforms are characterized by dynamic user behavior which produces highly variable and temporally rich traffic patterns. Unlike CNNs, which primarily capture local spatial features, RNNs are designed to model sequential dependencies over time. This makes them particularly better-suited for identifying temporal structures in traffic flows that result from bursty, user-driven interactions. The higher correct classification rates for TikTok and Instagram suggest that the RNN was

better able to leverage these temporal cues, effectively distinguishing them. This reinforces the advantage of using RNN-based architectures for traffic types where packet timing and flow order carry meaningful signals.

4.3 Autoencoder

The Autoencoder model is trained and evaluated using the Chalmers dataset, providing insights into its ability to reconstruct encrypted traffic data. The model's performance is reflected in the training and validation loss curves, as shown in Figure 4.3, and the loss values observed over the course of training.

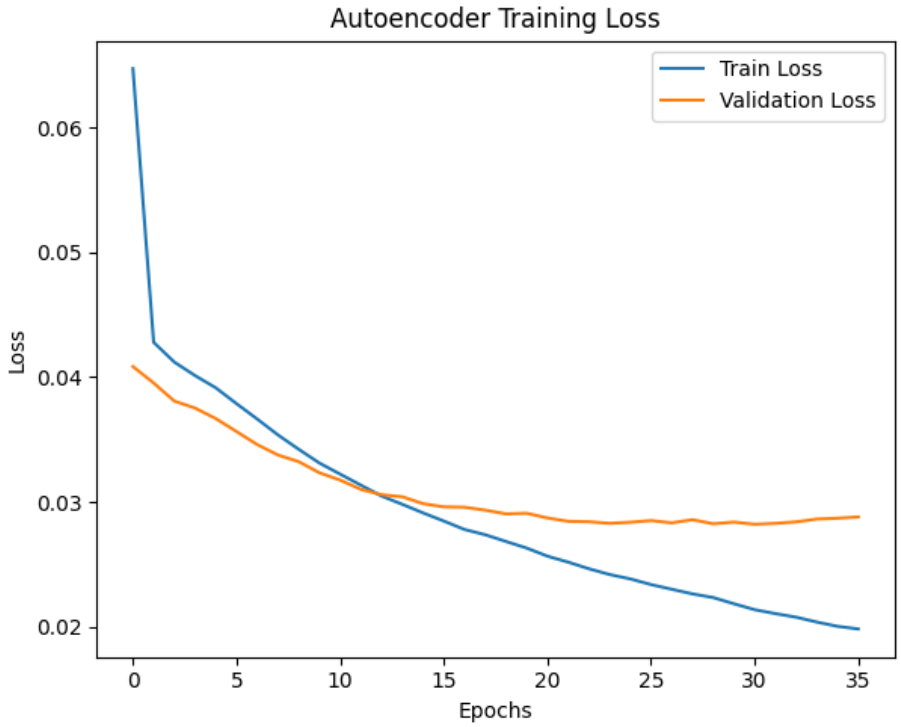

Figure 4.3: Training and validation loss curves for the Autoencoder model. The steady decrease in both losses indicates effective training and convergence.

Training Dynamics The training process involved 50 epochs, with a batch size of 32, and early stopping based on validation loss. The initial training loss started at 0.0647, while the validation loss was 0.0408. Over the epochs, both metrics steadily declined, with the training loss decreasing to 0.0198 and the validation loss stabilizing at 0.0288.

Loss Behavior The training and validation loss curves show a consistent downward trend during the early epochs, indicating effective learning of traffic patterns. By epoch 10, the validation loss had reached 0.0323, demonstrating the model's capa-

bility to generalize to unseen data. While there is a slight gap between the training and validation losses, the curves remain parallel, suggesting minimal overfitting.

Model Performance The final reconstruction loss values highlight the Autoencoder's effectiveness in compressing and reconstructing the input data. The smooth convergence of the loss curves underscores the stability of the training process. These results demonstrate that the model successfully captures the intrinsic structure of the encrypted traffic data, enabling its application for tasks like anomaly detection or feature extraction.

Autoencoder-Based Classification Approach: Autoencoders are typically designed for unsupervised learning, where they learn to encode and reconstruct input data. However, in this study, the AE was repurposed for feature extraction in a supervised classification pipeline as well. This was achieved by leveraging the encoders compressed representation of network traffic data, rather than using the AE for direct classification.

The AE is trained on network traffic data, where it learned a lower-dimensional latent representation that preserves essential characteristics while discarding redundant information. The network architecture consists of:

- An encoder which is responsible for compressing high-dimensional input data into a 128-dimensional feature space.
- A Decoder: Used to reconstruct the original input, ensuring that the encoder retains meaningful traffic patterns.

Once the Autoencoder is trained, the encoder output is extracted as a feature representation of the network traffic data. This transformed the input from a raw packet-based format into a compact, high-level feature space, which was then used for classification.

To perform classification, the extracted feature representations were fed into a supervised classifier. A Multi-Layer Perceptron (MLP) was selected for this task due to its effectiveness in handling structured feature sets. The classifier is trained using the encoded outputs as inputs and the corresponding application labels as targets.

Table 4.3 and Figure 4.4 below are the results produced by feeding the MLP the encoded VPN encrypted traffic data.

The classification results indicate that the Autoencoder model performs well in certain classes, particularly Spotify (*F1-score: 0.70*) and Youtube (*F1-score: 0.67*). These classes exhibit high recall values, suggesting that the model is effective in identifying instances of these categories. However, for other classes such as Instagram (*F1-score: 0.28*) and Twitter (*F1-score: 0.29*), the model struggles to achieve high precision and recall, indicating possible misclassification.

Table 4.3: Classification Report for Autoencoder Model over Encrypted Traffic

Class	Precision	Recall	F1-Score	Support
Facebook	0.31	0.35	0.33	86
Instagram	0.37	0.22	0.28	99
LinkedIn	0.35	0.68	0.46	80
Spotify	0.58	0.88	0.70	95
Tiktok	0.53	0.34	0.41	85
Twitter	0.41	0.22	0.29	89
Wikipedia	0.49	0.34	0.40	96
Youtube	0.65	0.68	0.67	82
Accuracy	0.46 (712 instances)			
Macro Avg	0.46	0.47	0.44	712
Weighted Avg	0.46	0.46	0.44	712

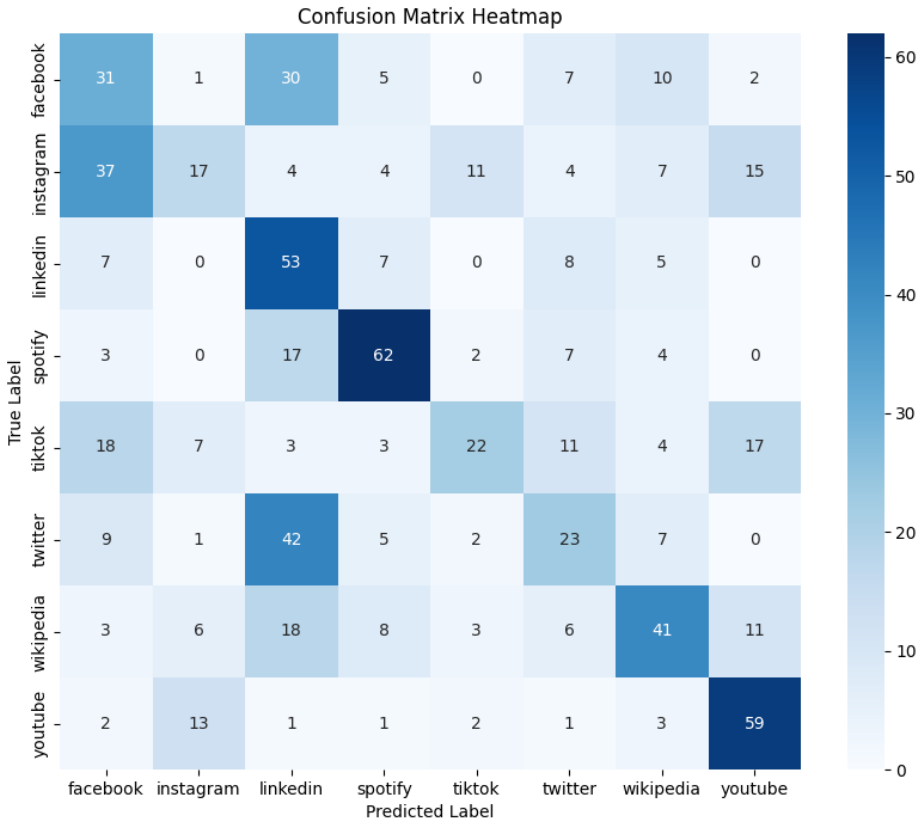

Figure 4.4: Confusion matrix illustrating the performance of the Autoencoder model on the Chalmers dataset. The matrix highlights true labels versus predicted labels across eight traffic classes.

4.4 Training, Inference, and Preprocessing Time Analysis

The performance of the implemented models was evaluated not only in terms of classification metrics but also regarding their computational efficiency. Specifically, the time required for training and inference was measured across the CNN, Autoencoder, and data preprocessing pipeline.

The CNN-based model demonstrated a total training time of approximately 10.64 seconds, with inference across the test set completing in 0.32 seconds. Given the architectural similarity and computational behavior of RNN-based models (especially in Keras implementations), which also involve sequential data and similar layer-wise operations (e.g., time-unrolled LSTM cells versus convolutional filters), the RNN is expected to exhibit comparable performance characteristics [7] however, training time and inference time were not measured in this study due to time constraints. Empirical results in prior literature [7] suggest that, with similar dataset sizes and batch processing configurations, RNN training times tend to fall within the same magnitude as CNNs, with minor deviations due to recurrent state updates.

In contrast, the Autoencoder architecture achieved a notably faster training time of 4.29 seconds. Its inference phase was similarly efficient, completing in 0.18 seconds. The lower computational demand is attributed to the simpler dense-layer structure of the Autoencoder.

The most significant reduction in execution time was observed in the data preprocessing phase, which saw a huge improvement from a previous runtime of approximately 6 hours and 30 minutes to just under 2.5 minutes (141.9 seconds). This optimization was achieved primarily in the data preprocessing step, which leverages the ET-BERT method to extract three key fields directly from traffic flows timestamp, packet length, and direction while avoiding the Wang et al. approach that involves converting raw packets into PNG images and subsequently transforming them into MNIST style tensors [8] [4]. The gain here was particularly impactful, as preprocessing forms a non-negligible component of the overall pipeline and often dictates the feasibility of experimentation and hyperparameter tuning.

4.5 Generalization to External Datasets

In addition to the Chalmers dataset used for model development and primary evaluation, two external datasets were incorporated to assess the generalization capability of the selected models: the MIT VPN dataset [45] and the UNB ISCX VPN dataset [46]. These datasets represent real-world encrypted VPN traffic from diverse applications and were selected to evaluate model robustness across different traffic distributions. However, due to their highly imbalanced class distributions and limited sample sizes in certain categories, they were not included in the main results section. The imbalance caused the models to overfit to dominant classes, resulting in unreliable evaluation metrics. Nevertheless, these experiments provide valuable

insight into model behaviour under practical but imperfect data conditions.

4.5.1 CNN on UNB

The CNN model was evaluated on the UNB dataset, which consists of encrypted VPN traffic from nine different applications spanning a range of protocols, including but not limited to FTP, VoIP, and TCP-based services. Due to the significant class imbalance present in the dataset, the model's predictions were lean heavily toward dominant classes such as Skype and VoIPBuster. While it achieved relatively high recall on these frequent classes, its performance on underrepresented applications like Email, ICQ, and Facebook was poor, resulting in an overall limited generalization capability. Below is an interpretation of the confusion matrix and classification metrics:

Table 4.4: Classification Report for CNN on UNB Dataset

Application	**Precision**	**Recall**	**F1-Score**	**Support**
Email	0.00	0.00	0.00	2
Facebook	0.00	0.00	0.00	1
FTPS	0.38	0.89	0.53	9
Hangout	0.00	0.00	0.00	24
ICQ	0.00	0.00	0.00	1
SFTP	0.00	0.00	0.00	6
Skype	0.38	0.77	0.50	43
Vimeo	1.00	0.65	0.79	17
VoIPBuster	0.50	0.24	0.33	37
Accuracy	0.44 (140 instances)			
Macro Avg	0.25	0.28	0.24	140
Weighted Avg	0.39	0.44	0.37	140

Precision: The model achieved perfect precision for the Vimeo class (1.00) and reasonable precision for Skype (0.38) and VoIPBuster (0.50), but performed poorly on all other classes, with zero precision.

Recall: FTPS had the highest recall (0.89), suggesting the model was effective at detecting its instances. However, most other classes such as Email, Facebook, and Hangout had zero recall, indicating complete failure in detection.

F1-Score: The highest F1-score was observed for Vimeo (0.79), while Skype and FTPS also showed moderate performance. All other classes had F1-scores of 0.00.

Accuracy: The model achieved an overall accuracy of 44%, reflecting its tendency to overfit to dominant classes due to the dataset's imbalance.

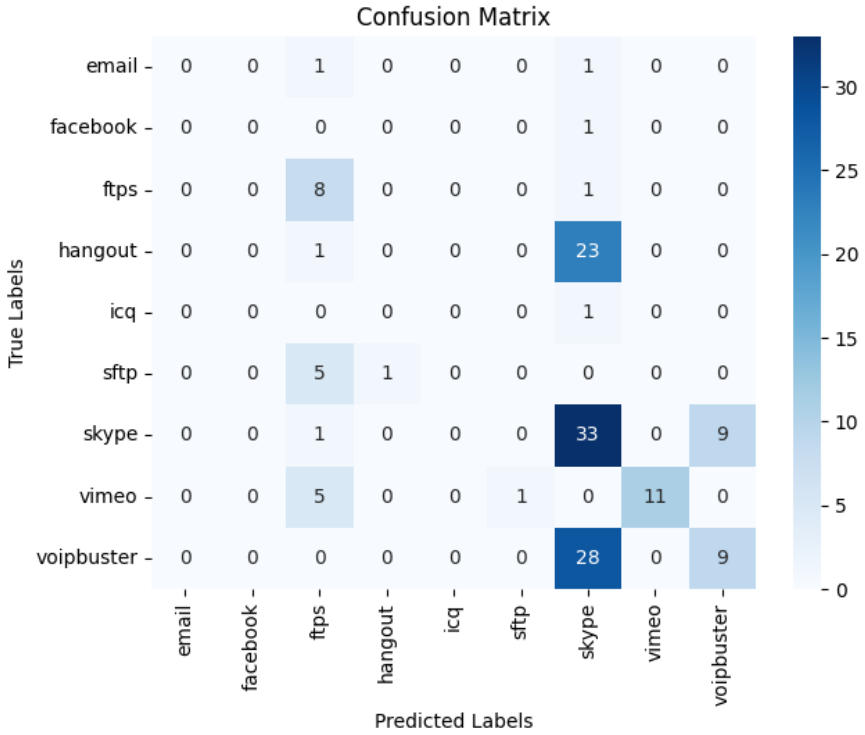

Figure 4.5: Confusion matrix illustrating the performance of the CNN model on the UNB dataset. The matrix highlights true labels versus predicted labels across nine traffic classes.

The confusion matrix in Figure 4.5 shows relatively strong predictive accuracy for FTPS, Skype, and Vimeo traffic, as indicated by higher values along the diagonal for these classes. However, it struggles with several minority classes, such as Email, Facebook, and ICQ, where predictions are often misclassified or absent entirely. The distribution of misclassifications suggests that the CNN is biased toward dominant classes with more consistent or distinguishable traffic patterns, while its performance degrades on underrepresented application flows.

4.5.2 Autoencoder on UNB

The Autoencoder model is trained and evaluated using the UNB dataset, providing insights into its ability to reconstruct encrypted traffic data and extract representative latent features. The models performance is reflected in the training and validation loss curves, as shown in Figure 4.6, and the classification outcomes discussed below.

Training Dynamics The training process was conducted over 50 epochs, using a batch size of 32. The initial training loss was approximately 0.157, while the validation loss started at around 0.087. Both losses declined steadily throughout training. By the end of the process, the training loss had dropped to approximately 0.016 and the validation loss stabilized around 0.024.

Loss Behavior As shown in Figure 4.6, the training and validation loss curves follow a consistent downward trend, with minimal gap between them, indicating a well-generalized model and the absence of significant overfitting.

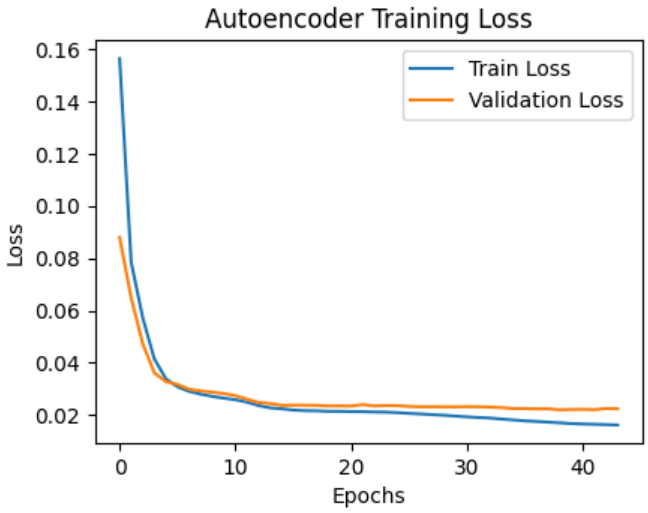

Figure 4.6: Training and validation loss curves for the Autoencoder model on the UNB dataset. The steady decrease in both losses indicates effective training and convergence.

Model Performance The final reconstruction and classification performance highlight the Autoencoders ability to capture essential structures in encrypted traffic.

The following section outlines how its latent output was used for supervised classification.

4.5.2.1 Autoencoder-Based Classification Approach on UNB

The compressed representations from the encoding phase were used as inputs to a supervised classification pipeline. Specifically, MLP is trained using the encoded outputs and their corresponding class labels. The classification results are shown in Table 4.5 and Figure 4.7.

The classification results indicate that the Autoencoder model performs well in several classes, particularly VoIPBuster (F1-score: 0.75), Vimeo (0.79), and Hangout (0.74). These applications exhibit high recall, showing the models ability to accurately detect most of their instances. However, minority classes such as Email, Facebook, and ICQ have zero F1-scores, reflecting insufficient data for reliable classification.

Table 4.5: Classification Report for Autoencoder Model on UNB Dataset

Application	Precision	Recall	F1-Score	Support
Email	0.00	0.00	0.00	2
Facebook	0.00	0.00	0.00	1
FTPS	0.36	0.89	0.52	9
Hangout	0.84	0.67	0.74	24
ICQ	0.00	0.00	0.00	1
SFTP	0.00	0.00	0.00	6
Skype	0.83	0.56	0.67	43
Vimeo	1.00	0.65	0.79	17
VoIPBuster	0.61	0.97	0.75	37
Accuracy	0.68 (140 instances)			
Macro Avg	0.40	0.41	0.38	140
Weighted Avg	0.70	0.68	0.66	140

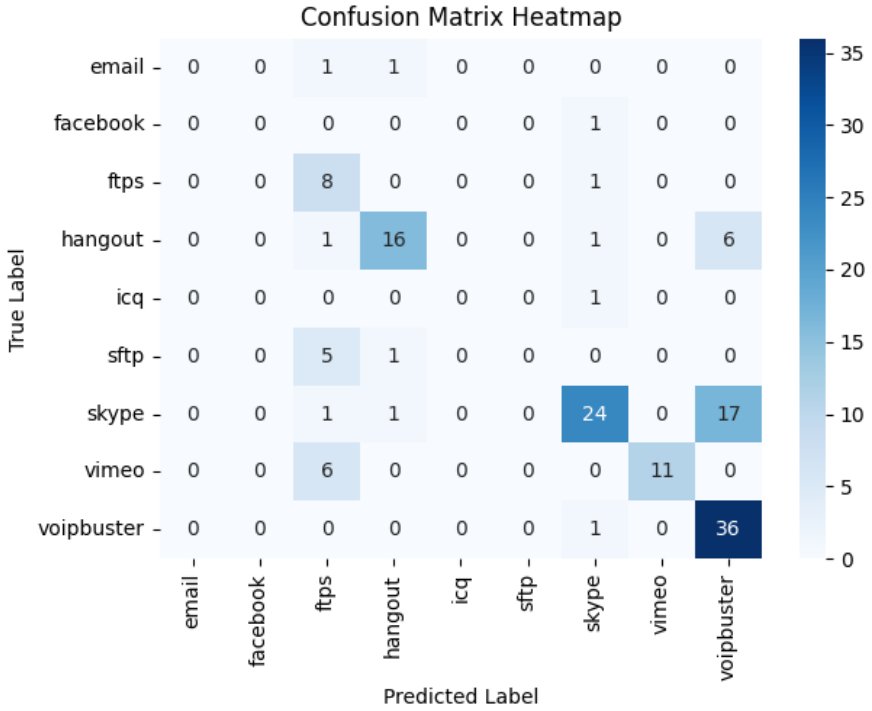

Figure 4.7: Confusion matrix illustrating the performance of the Autoencoder model on the UNB dataset. The matrix highlights true labels versus predicted labels across nine traffic classes.

4.5.3 CNN on MIT

The CNN model is trained and evaluated using the MIT dataset [45], which contains VPN-encrypted traffic from eight different applications. The results provide insight into how the model handles imbalanced datasets with dominant and minority classes. Below is an interpretation of the confusion matrix and classification metrics:

Table 4.6: Classification Report for CNN on MIT Dataset

Application	Precision	Recall	F1-Score	Support
RDP	0.00	0.00	0.00	1
RSYNC	0.00	0.00	0.00	1
SCP	0.00	0.00	0.00	1
SFTP	0.00	0.00	0.00	2
Skype	0.76	1.00	0.86	28
SSH	0.00	0.00	0.00	2
VoIP	0.00	0.00	0.00	1
YouTube	0.00	0.00	0.00	1
Accuracy	0.76 (37 instances)			
Macro Avg	0.09	0.12	0.11	37
Weighted Avg	0.57	0.76	0.65	37

Table 4.6 summarizes the classification performance across eight classes of VPN-encrypted traffic. The metrics reveal the model's high confidence in classifying the dominant class, Skype, but an inability to detect minority classes.

Precision: The model achieved a high precision of 0.76 for Skype but scored 0.00 for all other applications, indicating a strong bias toward the majority class and a high rate of false positives for underrepresented categories.

Recall: Skype was perfectly recalled with a score of 1.00, while the recall for all other classes was 0.00, confirming that the model failed to identify any instances of minority classes such as RDP, RSYNC, and VoIP.

F1-Score: The F1-score for Skype was 0.86, while all other classes scored 0.00, emphasizing the severe impact of class imbalance on model performance.

Accuracy: The overall accuracy was 76%, which is misleadingly high due to the over-representation of Skype in the dataset.

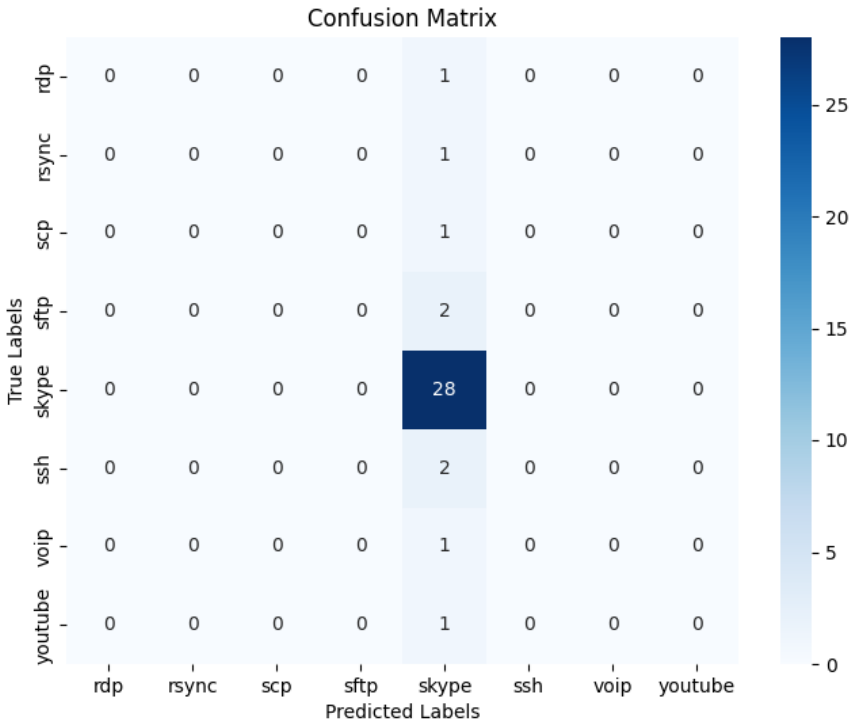

Figure 4.8: Confusion matrix illustrating the performance of the CNN model on the MIT dataset. The matrix highlights true labels versus predicted labels across eight traffic classes.

The confusion matrix in Figure 4.8 shows how the CNN model performed across the eight application classes in the MIT dataset [45]. The model correctly classified most of the Skype traffic because it had a bigger chunk of training data, making its patterns easier to learn. On the other hand, all other applications, which had fewer examples, were misclassified. This suggests that the model tended to focus on the types of traffic it saw most frequently and struggled with classes that were less familiar or showed more variation.

4.5.4 Autoencoder on MIT

The Autoencoder model is trained and evaluated using the MIT dataset, which contains encrypted VPN traffic across a range of applications. This section presents the training behavior and classification performance of the model when applied to this imbalanced dataset.

Training Dynamics The training process lasted for 50 epochs using a batch size of 32. Initially, the training loss was around 0.183, and the validation loss began at approximately 0.150. These values declined rapidly in the first few epochs. By the final epoch, the training loss had reduced to approximately 0.022, while the validation loss plateaued near 0.024.

Loss Behavior Figure 4.9 shows the training and validation loss curves. The

consistent drop in both metrics and the close proximity between them indicate effective convergence and minimal overfitting throughout the training process.

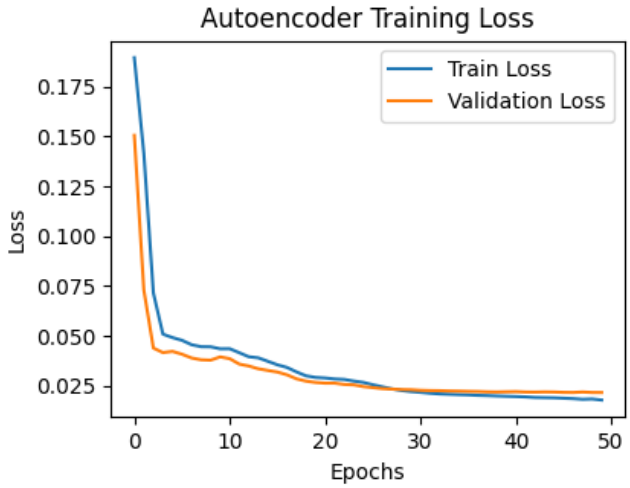

Figure 4.9: Training and validation loss curves for the Autoencoder model on the MIT dataset.

Model Performance After training, the encoder outputs were used for classification by an MLP. The goal was to assess the Autoencoders capacity to generalize traffic characteristics from its latent representations. The classification results are presented in Table 4.7 and Figure 4.10.

4.5.4.1 Autoencoder-Based Classification Approach on MIT

Table 4.7: Classification Report for Autoencoder Model on MIT Dataset

Application	Precision	Recall	F1-Score	Support
RDP	0.00	0.00	0.00	1
RSYNC	0.33	1.00	0.50	1
SCP	0.00	0.00	0.00	1
SFTP	0.00	0.00	0.00	2
Skype	0.82	1.00	0.90	28
SSH	0.00	0.00	0.00	2
VoIP	0.00	0.00	0.00	1
YouTube	0.00	0.00	0.00	1
Accuracy	0.78 (37 instances)			
Macro Avg	0.14	0.25	0.18	37
Weighted Avg	0.63	0.78	0.70	37

The classification results indicate that the Autoencoder model performs exceptionally well for Skype (F1-score: 0.90), the dominant class in the dataset. While RSYNC also achieved 100% recall due to its single sample, all other classes received

zero scores, reflecting the severe class imbalance and limited sample availability for minority applications.

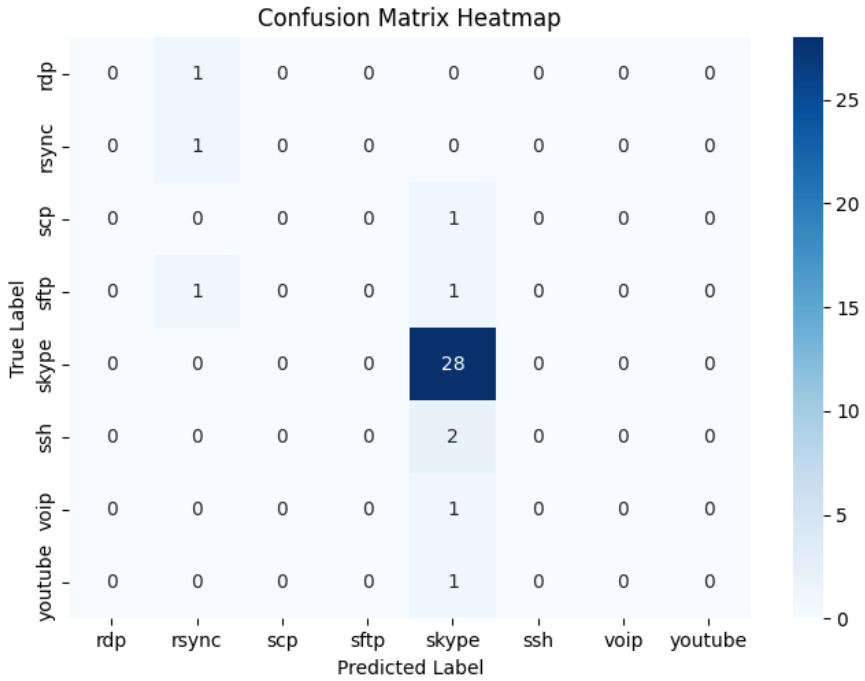

Figure 4.10: Confusion matrix illustrating the performance of the Autoencoder model on the MIT dataset. The matrix highlights true labels versus predicted labels across eight traffic classes.

4.5.5 Training, Inference, and Preprocessing Time Analysis

To evaluate the practical feasibility of the proposed models, we measured training and inference times across both the MIT and UNB datasets. These metrics provide insight into the computational demands of each architecture and the potential trade-offs between accuracy and efficiency.

MIT Dataset: On the MIT dataset, both models completed training quickly due to the small dataset size (37 samples). The CNN model required approximately **3.71 seconds** for training and **0.22 seconds** for inference. The Autoencoder model trained in **2.73 seconds** and completed inference in **0.12 seconds**. Preprocessing for both models, including normalization and label encoding, took less than **1 second**, primarily due to the compact input size.

UNB Dataset: For the larger UNB dataset (140 samples), training and inference times were proportionally longer. The CNN model trained in **7.65 seconds** and performed inference in **0.25 seconds**. In contrast, the Autoencoder model completed training in **4.79 seconds** and inference in **0.15 seconds**. Preprocessing for the UNB dataset took approximately **1.2 seconds**, due to higher input dimensionality and increased sample count.

Across both datasets, the Autoencoder model demonstrated not only improved

classification performance but also faster training and inference times compared to the CNN. **This suggests that Autoencoder-based architectures offer a favourable balance between performance and computational cost**, making them suitable candidates for real-time or resource-constrained scenarios.

4.6 Summary of Results

This section presents a comparative summary of the model performances across the three datasets used in this study: Chalmers [44], MIT [45], and UNB [46] . Each model CNN, RNN, and Autoencoder was evaluated using standard classification metrics such as accuracy, weighted F1-score, and macro F1-score. The results are consolidated in Table 4.8, which highlights performance variations across datasets and model architectures.

Table 4.8: Summary of Model Performance Across Datasets

Dataset	Model	Accuracy	Weighted F1	Macro F1	Dominant Classes
3*Chalmers [44]	CNN	52%	0.51	0.51	YouTube
	RNN	49%	0.47	0.45	Facebook, Wikipedia
	Autoencoder	46%	0.44	0.44	Spotify, YouTube
2*MIT [45]	CNN	76%	0.65	0.11	Skype
	Autoencoder	78%	0.70	0.18	Skype, RSYNC
2*UNB [46]	CNN	44%	0.37	0.24	Skype
	Autoencoder	68%	0.66	0.38	VoIPBuster, Vimeo

As seen in Table 4.8, the Autoencoder model consistently achieved higher performance than the CNN model across both external datasets (MIT and UNB), especially in terms of F1-scores and recall on minority classes. While the CNN demonstrated strong performance when trained on balanced data, its performance degraded on imbalanced datasets. The RNN, evaluated only on the Chalmers dataset, showed moderate results but lacked the generalization power exhibited by the Autoencoder.

4.7 Discussion

From our experiments, we found that CNNs and RNNs show strong potential in identifying traffic patterns despite encryption. CNNs perform well in feature extraction but struggle with sequential dependencies, whereas RNNs, particularly LSTM models, proved more effective in recognising patterns over time, leading to higher recall rates.

These findings underscore the importance of **model selection** and **dataset characteristics** in the context of VPN-encrypted traffic classification. Models that can learn robust latent representations, such as Autoencoders, offer significant advantages when applied to real-world, imbalanced data.

4.7.1 Challanges

Despite these promising results, the study also revealed key challenges, which are discussed below.

One of the primary limitations encountered, especially in the MIT and UNB datasets, is severe class imbalance. Most models, especially CNNs, exhibited a strong bias toward dominant classes such as Skype, while consistently failing to detect underrepresented classes like RSYNC, ICQ, or Facebook. The Autoencoder-based approach showed a comparatively better ability to generalise under imbalance, particularly on the UNB dataset. However, even this model suffered performance drops when class support fell below a minimum threshold. Future work could incorporate class-balancing strategies such as SMOTE [47], adaptive loss functions, or cost-sensitive learning to mitigate this issue.

Another challenge was the limited representation of application behaviours across datasets. The MIT dataset lacks clarity on whether the traffic originated from mobile or desktop devices, while the Chalmers dataset although balanced and carefully constructed includes only eight mobile applications. This restricts the models ability to generalise to broader, real-world traffic patterns. Expanding the training corpus to include more heterogeneous traffic sources from different platforms and usage contexts would improve model robustness and relevance.

Computational efficiency is another important consideration. While DL models improved classification performance, they also introduced significant computational overhead, particularly the RNN-based models. In contrast, the Autoencoder architecture achieved a favourable trade-off: it offered better generalisation than the CNN and completed training and inference faster across both the MIT and UNB datasets. These results suggest that Autoencoder-based models are viable for real-time or resource-constrained deployments, though further optimisation and testing in live environments would be necessary for practical integration.

Another challenge identified is the similarity of traffic patterns across certain application classes. Social media platforms such as Facebook and Instagram exhibited overlapping traffic behaviors, often leading to misclassification. This indicates a limitation in current model granularity and suggests that future approaches could benefit from hybrid architectures (e.g., CNN+RNN) or ensemble methods to better distinguish subtle behavioral differences between encrypted traffic classes.

Although encrypted traffic classification offers clear benefits for network management and security, it also raises ethical concerns. Even without decrypting payloads, metadata-based classification can potentially be misused for profiling or surveillance. Ensuring ethical use of these models requires integration with privacy-preserving machine learning frameworks, such as federated learning or differential privacy, to safeguard user anonymity while maintaining model effectiveness.

The results confirm that model performance is closely tied to data quality, class balance, and feature representation. Autoencoders, when combined with feature-rich preprocessing like ET-BERT, offer a robust and computationally efficient solution for classifying encrypted VPN traffic. However, real-world deployment requires ad-

dressing limitations related to dataset diversity, performance under imbalance, and privacy assurance. Continued exploration in these areas is essential to developing scalable, trustworthy traffic classification systems.

5

Conclusion

This study set out to evaluate the performance of deep learning models in classifying VPN-encrypted traffic, a task that has become increasingly relevant as more users rely on VPNs for privacy and security. Traditional classification methods that depend on port numbers or deep packet inspection have become ineffective due to encryption, prompting the need for more advanced techniques like deep learning.

This chapter concludes the thesis by summarizing the research objectives, key findings, and contributions, and by outlining the limitations of the study and potential directions for future work.

This thesis explored the use of DL models for classifying VPN-encrypted mobile traffic, addressing the challenge posed by encryption in traditional traffic classification methods. Three model architectures CNNs, RNNs, and Autoencoders were evaluated using both balanced and imbalanced real-world datasets.

The results demonstrate that while CNNs and RNNs are effective for extracting spatial and temporal features respectively, the Autoencoder offered the best trade-off between generalization performance and computational efficiency, especially under class imbalance. These findings highlight the potential of unsupervised or semi-supervised architectures in encrypted traffic classification scenarios.

Moreover, the study reinforces the importance of dataset design. Balanced datasets like the one from Chalmers enable more reliable benchmarking, while imbalanced datasets such as MIT and UNB reflect real-world challenges. Evaluation metrics including accuracy, F1-score, and inference time provided a comprehensive understanding of model behavior under different conditions.

A key insight is the potential of combining CNNs for spatial feature extraction and RNNs for temporal pattern modeling into a hybrid architecture. While not implemented in this work, this direction is motivated by the strengths observed in both models and should be investigated in future research.

Limitations of this study include the use of limited application classes, dataset imbalance in external sources, and absence of real-time deployment testing. Future work should focus on larger, more diverse traffic sources, continual learning mechanisms, and privacy-preserving training strategies.

In summary, this work contributes a comparative study of deep learning models for VPN-encrypted traffic classification, proposes promising directions for future work,

and provides insight into the design and deployment of secure, efficient, and accurate traffic classifiers.

Bibliography

[1] X. Wang, S. Chen, and J. Su, "App-net: A hybrid neural network for encrypted mobile traffic classification," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 424–429. DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162891.

[2] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1962–1976, 2021. DOI: 10.1109/TNSM.2021.3052888.

[3] S. Zhang, "An overview of network slicing for 5g," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019. DOI: 10.1109/MWC.2019.1800234.

[4] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019. DOI: 10.1109/ACCESS.2019.2912896.

[5] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, p. 106 944, 2019, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.106944. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619304669.

[6] M. Al-Dulaimi, S. A. Rahman, H. Al-Khateeb, and M. Rajarajan, "Encrypted vpn traffic classification: A machine learning approach," *IEEE Access*, vol. 12, pp. 67 890–67 905, 2024. DOI: 10.1109/ACCESS.2024.3352227. [Online]. Available: https://ieeexplore.ieee.org/document/11091298.

[7] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019. DOI: 10.1109/TNSM.2019.2899085.

[8] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the ACM Web Conference 2022*, Lyon, France, 2022. DOI: 10.1145/3485447.3512217.

[9] N. I. of Standards and T. (NIST), "Implementing a virtual private network (vpn)," 2021, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-21.pdf.

[10] CartoonDealer, *Vpn diagram vector illustration*, https://cartoondealer.com/image/142602777/vpn-diagram-vector-illustration-outline-virtual-private-network-lan-scheme.html, 2023.

[11] GeeksforGeeks, "Types of virtual private network (vpn) and its protocols," 2023, https://www.geeksforgeeks.org/types-of-virtual-private-network-vpn-and-its-protocols/.
[12] P. A. Networks, "Types of vpn protocols," 2023, https://www.paloaltonetworks.com/cyberpedia/types-of-vpn-protocols.
[13] NordLayer, "Vpn types and protocols," 2023, https://nordlayer.com/learn/vpn/types-and-protocols/.
[14] NordVPN, "Vpn protocols explained," 2023, https://nordvpn.com/blog/protocols/.
[15] M. Electronics, "Challenges in vpn traffic classification," *Electronics*, vol. 12, no. 1, p. 115, 2023, https://www.mdpi.com/2079-9292/12/1/115.
[16] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *National Science Foundation*, 2019, https://par.nsf.gov/servlets/purl/10097233.
[17] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*. Nature Publishing Group, 2015, vol. 521, pp. 436–444.
[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.
[19] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.
[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," pp. 1097–1105, 2012.
[21] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," pp. 5998–6008, 2017.
[22] J. Jumper, R. Evans, A. Pritzel, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.
[23] M. Reichstein, G. Camps-Valls, B. Stevens, *et al.*, "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.
[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[25] G. Litjens, T. Kooi, B. E. Bejnordi, *et al.*, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.
[26] M. Bojarski, D. Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
[27] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," pp. 815–823, 2015.
[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[29] K. Cho, B. Van Merri"enboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," pp. 1724–1734, 2014.
[30] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," pp. 6645–6649, 2013.

[31] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, 2018.

[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[33] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[34] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," pp. 1096–1103, 2008.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[36] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[37] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4–11, 2014.

[38] A. Balachandran and P. P. Amritha, "Vpn network traffic classification using entropy estimation and time-related features," in *IOT with Smart Systems*, T. Senjyu, P. Mahalle, T. Perumal, and A. Joshi, Eds., Singapore: Springer Nature Singapore, 2022, pp. 509–520, ISBN: 978-981-16-3945-6.

[39] A. A. Afuwape, Y. Xu, J. H. Anajemba, and G. Srivastava, "Performance evaluation of secured network traffic classification using a machine learning approach," *Computer Standards & Interfaces*, vol. 78, p. 103 545, 2021.

[40] Y. Ma, Z. Li, H. Xue, and J. Chang, "A balanced supervised contrastive learning-based method for encrypted network traffic classification," *Computers Security*, vol. 145, p. 104 023, 2024, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2024.104023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824003286.

[41] A. Saber, B. Fergani, and M. Abbas, "Encrypted traffic classification: Combining over-and under-sampling through a pca-svm," in *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, 2018, pp. 1–5. DOI: 10.1109/PAIS.2018.8598480.

[42] T. Shapira and Y. Shavitt, "Flowpic: A generic representation for encrypted traffic classification and applications identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1218–1232, 2021. DOI: 10.1109/TNSM.2021.3071441.

[43] A. Balachandran and P. P. Amritha, "Vpn network traffic classification using entropy estimation and time-related features," in *IOT with Smart Systems*, T. Senjyu, P. Mahalle, T. Perumal, and A. Joshi, Eds., Singapore: Springer Nature Singapore, 2022, pp. 509–520, ISBN: 978-981-16-3945-6.

[44] E. Nehme, K. Fitzgerald, and A. Kassler, *5g mobile app traffic traces - chalmers 2023*, https://ieee-dataport.org/documents/5g-mobile-app-traffic-traces-chalmers-2023, Accessed: 2025-06-01, 2023.

[45] *Vpn/non-vpn network application traffic dataset (vnat)*, https://www.ll.mit.edu/r-d/datasets/vpnnonvpn-network-application-traffic-dataset-

`vnat`, Lincoln Laboratory, Massachusetts Institute of Technology, Accessed: 2024.

[46] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, Rome, Italy, 2016, pp. 407–414.

[47] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

A

Appendix 1