UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Classifying Encrypted WiFi Traffic Us ing Deep Learning Methods

by

Ying Li
Supervisor: Dr. Christy Jie Liang

Thesis for
Doctor of Philosophy

Sydney, Australia

2022

Certificate of Original Authorship

I, Ying Li declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature:

Date: 1/12/2022

ABSTRACT

Classifying Encrypted WiFi Traffic Using Deep Learning Methods

by

Ying Li

Supervisor: Dr. Christy Jie Liang

In this thesis, the goal is to classify encrypted WiFi traffic using deep learning methods.

1) Firstly, we investigate the possibility of making useful inferences from passively observed WiFi traffic that is encrypted at both the transport layer as well as the MAC layer. This is more challenging in comparison to making predictions from the IP layer traffic due to the lack of any meta information. We identify content from encrypted network traffic flows using video streaming as an example because videos are highly popular on the Internet and are frequently misused in many ways including the distribution of fake news, hate speech, and radical and propaganda content. Besides, in network protection and situational awareness applications, there is a strong need to identify whether certain known videos are being watched, either by certain individuals or in a certain area. In the first work, we create a video wireless traffic dataset that contains 10 YouTube videos collected at the WiFi layer. And we demonstrate the possibility of identifying video content using different deep learning models. We do experiments on this traffic dataset and show that our model can achieve a good performance. Besides, we evaluate the longevity of our classifier by making predictions two weeks apart. The results of this work will be further elaborated in Chapter Three.

2) Secondly, not limited to video streaming, other types of traffics(e.g. web and

audio streaming) need to do classification due to the purpose of service management. However, only a limited amount of work has looked into the possibility of building a generic traffic classifier that can handle different classes of traffic. we show that encrypted WiFi traffic fingerprinting can be generalized and applies to many common internet traffic types such as web, video streaming, and audio streaming. In this work, we expand our video wireless traffic dataset to a general wireless traffic dataset that includes web, video streaming, and audio streaming. And we propose a novel hierarchical classifier that can make coarse-grained predictions (e.g. web, video, or audio) as well as fine granular predictions (e.g. content providers/platforms and exact content). Moreover, this approach allows us to estimate network usage characteristics for the purpose of service management in large networks and also identify unknown service providers for different traffic classes. This is explained in detail in Chapter Four.

3) Finally, we investigate how to generate WiFi traffic samples by category automatically. A high-quality, high-volume dataset is very important for the deep learning-based classifier. Specific to the network domain, the classifier is sensitive to the dataset. For example, the network environment of an individual and an enterprise is different in terms of network transfer speed and network configures. Besides, data collection is time-consuming. Therefore, a generator that can generate samples by category automatically is needed. There are many existing generative models. But, the labeled data is required when they generate samples by category. In this work, we propose two novel generative models, namely infinite Gaussian mixture auto-encoder(IGMVAE) and the infinite mixture of infinite Gaussian mixture auto-encoder (I^2GMVAE). IGMVAE is a variant of variational auto-encoder(VAE) with an infinite Gaussian Mixture model (IGMM) as the prior distribution of the latent variables. I^2GMVAE is a variant of VAE with the infinite mixture of infinite Gaussian Mixture model (I^2GMM) as the prior distribution of the latent variables. They are explained in detail in Chapter Five.

Acknowledgements

I would like to thank the following people. They give me great help and support during my Ph.D. journey.

First and foremost I wish to thank my supervisor, Dr. Christy Jie Liang. It is my great luck to meet her as my supervisor. Christy is very professional and kind. She not only inspired me academically but also help me a lot in life. Her good personality will affect me in my future life.

I would like to thank my co-supervisor, Prof. Richard Yida Xu. Richard is not only a professional scholar but also a person who loves to share knowledge. He gives us weekly courses on machine learning and computer technology. The most amazing thing is that he can let us learn a lot of complicated knowledge in an easy-to-understand way. His hardworking and kindness will benefit me for life.

I would also like to thank my co-supervisor, Prof. Massimo Piccardi. He helped me a lot. Thanks so much for your patience and kindness.

I also want to thank Dr. Junyu Xuan. Junyu is a very professional and responsible scholar. We discussed the innovations, model details, and experiments together. It is my pleasure to have a such good partner in the past few years.

I would also like to thank my research partners, including Suranga Seneviratne, Guillaume Jourjon, Adriel Cheng, Darren Webb, Kanchana Thilakarathna, and David B. Smith. We share knowledge and discuss experiment details every week. Thank you for giving me an unforgettable and rewarding project experience.

I am grateful to all lab mates, including Yi Huang, Shuai Jiang, Wanming Huang, Haodong Chang, Caoyuan Li, Xuan Liang, Jason Traish, Ziyue Zhang, Wei Huang, Chen Deng, Congzhentao Huang, Chris Markos, Yunce Zhao, etc. We shared knowledge and hiked together on weekend. Because of those mates, I had a pleasant and meaningful time in Sydney.

Thanks to the CA panel Prof. Guandong Xu and Prof. Andrew Zhang.

Sincerely, I want to thank my parents, my sister, my husband, and my child. Because of your support and encouragement, I can concentrate on my research and move forward.

Thanks again to everyone who has helped me!

List of Publications

Conference Papers

C-1. **Ying Li**, Yi Huang, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Guillaume Jourjon, Darren Webb and Richard Yi Da Xu, "Deep Content: Unveiling Video Streaming Content from Encrypted WiFi Traffic", *17th Int. Symp. on Network Computing and Applications – NCA 2018.* **(CORE A conference)**

Journal Papers

J-1. **Ying Li**, Yi Huang, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Guillaume Jourjon, Darren Webb, David B. Smith and Richard Yi Da Xu, "From Traffic Classes to Content: A Hierarchical Approach for Encrypted Traffic Classification", *Computer Networks* **(CORE B journal)**

J-2. **Ying Li**, Junyu Xuan, Yi Huang, Christy Liang and Richard Yida Xu, "Infinite Gaussian Mixture Autoencoders for Data Generation",*Transactions on Image Processing***(ready to submit)**

J-3. Yi Huang, **Ying Li**, Timothy Heyes, Guillaume Jourjon, Adriel Cheng, Suranga Seneviratne, Kanchana Thilakarathna, Darren Webb and Richard Yi Da Xu, "Probability Based Task Adaptive Siamese Open-Set Recognition for Encrypted Network Traffic With Bidirectional Dropout Data Augmentation", *Pattern Recognition Letters* **(CORE B journal)**

J-4. Yi Huang, **Ying Li**, Guillaume Jourjon, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb and Richard Yi Da Xu, "CRAAE:

Calibrated Reconstruction Based Adversarial AutoEncoder Model for Novelty Detection", *Pattern Recognition Letters* **(Under Review, CORE B)**

Contents

List of Figures

List of Tables

Abbreviation

TLS - Transport Layer Security
WPA2 - WiFi Protected Access 2
VAE - Variational Auto-encoder
IGMVAE - Infinite Gaussian Mixture Auto-encoder
I^2GMVAE - Infinite Mixture of infinite Gaussian Mixture Auto-encoder
IGMM - Infinite Gaussian Mixture Model
I^2GMM - Infinite Mixture of Infinite Gaussian Mixture Model
HTTPS - Hypertext Transfer Protocol Secure
ML - machine learning
DL - Deep learning
MLP - Multi-Layer Perceptron
CNN - Convolutional Neural Network
RNN - Recurrent Neural Network
LSTM - Long Short-term Memory
ABC - Australian Broadcasting Corporation
SMH - Sydney Morning Herald
GANs - Generative Adversarial Nets
TCP -Transmission Control Protocol
P2P - Peer-to-peer Internet
AAE - Adversarial Auto-encoder
DASH - Dynamic Adaptive Streaming over HTTP
HAS - HTTP based Adaptive Streaming
GMM - Gaussian Mixture Model
GMVAE - Gaussian Mixture Auto-encoder
BNP - Bayesian Nonparametric
ELOB - Evidence Lower Bound

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Encrypted WiFi Traffic Analysis

With the increasing scale of the network, the increasing variety of services, the increasing number of users, and the increasingly complex network behavior, how to effectively classify network traffic is an important issue for all network managers. More than 95% websites are end-to-end encrypted [62; 38]. While Transport Layer Security (TLS), the underlying security protocol behind HTTPS, provides confidentiality for the message that is being exchanged between two parties, the wide adoption of end-to-end encryption opens up several issues in terms of network management. For example, end-to-end encryption eliminates the possibility of traffic analysis in the core network for intrusion detection and parental filtering. It also hinders network optimizations carried out by telecommunication operators. Various parties are using this limitation to distribute problematic content such as fake news, copy-righted material, and propaganda videos.

Previous research explored the possibility of making inferences based on encrypted traffic flows by capturing network data packets at the IP layer [97; 4; 25]. This is viable because despite the message content being encrypted, the statistical properties of traffic flow such as packet lengths, inter-packet times, burst sizes, and burst intervals can still reveal information about the underlying encrypted traffic that is in transit. Also, at the IP level, there are other useful meta-data such as IP addresses, ports, and TLS header information.

In this thesis, we investigate the possibility of making useful inferences from passively observed WiFi traffic that is encrypted at both the TLS and MAC layers. This is more challenging in comparison to making predictions from the IP layer traffic due to the lack of any meta information. As such, there is increasing interest in making inferences and predictions from encrypted traffic flows.

1.1.2 Deep Learning based Traffic Classification Technology

Traditional traffic classification methods(e.g. port-based and payload-based methods) have some practical problems, such as dynamic ports and the widespread use of Hypertext Transfer Protocol Secure (HTTPS). Thus, traffic classification technology based on machine learning becomes the mainstream. It uses the statistical characteristics of network flow to identify traffic content. These features can be packet based or network flow based. The statistical characteristics based on data packets take the data packets in the network flow as the research object, mainly involving the size of packets, arrival interval, rate, etc. The statistical characteristics based on network flow take the network flow as the research object, and its characteristics mainly include flow size (e.g. number of packets, number of bits), duration, number of flag bits, etc. These methods are mainly used in application classification. However, there is a strong need to do content classification. For example, whether specific content has been watched by a specific person. Deep learning (DL) methods can incorporate feature learning into the process of building models, which can express more complex information and more completely interpret the relationship between features. It is more suitable for content identification.

1.1.3 Hierarchical Architecture for Traffic Classification

There are various types of traffic flows in real life and they can come from different providers. There is a need to build a generic traffic classifier that can handle different types of traffic such as web, video streaming, and audio streaming. A hierarchical architecture is able to exploit commonality in traffic from the same data provider, but also cater for dedicated classifiers for poorly separated traffic categories. Furthermore, a hierarchical approach also allows the estimation of network usage characteristics for the purposes of service management in large networks, which constitutes a first step toward identifying unknown service providers for different traffic classes.

1.1.4 Data Augmentation Technology

The quality and size of the dataset are important factors for the performance of deep learning based classifiers. Specific to the network domain, the traffic classifier is very sensitive to the network environment in which the data is captured. For example, the traffic flows for the same content vary widely in different environments. And it is also very sensitive to timeliness. There are also some differences in the same content traffic of different periods. However, in most cases, we capture data in a relatively single environment and over the same period. Therefore, how to automatically generate high-quality samples is an important topic. Data augmentation is a technique that expands a training dataset by generating more equivalent data from limited data. It is an effective method to overcome the lack of training data and is currently widely used in various fields of deep learning. There are many data argument methods, amount of which generative models are commonly used. This thesis proposed two novel generative models that can generate high-quality samples by categories automatically.

1.2 Research Objectives

The vast majority of Internet traffic is now end-to-end encrypted. While encryption provides user privacy and security, it has made network surveillance an impossible task. Recent advances in machine learning techniques have shown great promise in extracting content fingerprints from encrypted traffic captured at various points in IP core networks. Nonetheless, content fingerprinting from listening to encrypted wireless traffic is a challenging task due to the difficulty in distinguishing re-transmissions and multiple flows on the same link. Therefore, in the thesis, we investigate how to classify WiFi traffic using deep learning methods.

Through this thesis, we show the potential of fingerprinting internet traffic by passively sniffing WiFi frames in air, without connecting to the WiFi network by leveraging deep learning methods. In addition, we construct a hierarchical model to perform a classification of various types of traffic data. Besides, we investigate how to automatically generate training samples by category using deep learning methods. This thesis proposes to achieve the following three research objectives:

Research Objective ①: Classifying encrypted WiFi videos using deep learning models

Videos are highly popular and are frequently misused in many ways that include distribution of fake news, hate speech, and radical and propaganda content[37; 21; 23]. In network protection and situational awareness applications, there is a strong need to identify whether certain known videos are being watched, either by certain individuals or in a certain region. We designed three deep learning methods for encrypted WiFi videos classification: (a) Multi-Layer Perceptron (MLP). We selected multiple features as shown in Chapter 3. If only one feature has been chosen, the input size is 1*500. In this case, we can explore feeding this vector into multiple fully-Connect layers. (b) Convolutional Neural Network (CNN). Schuster et al. [97] apply a CNN model to classify the content of the traffic. This is similar to the goal of our work. The CNN model used in [97] reported excellent performance, and we implement this model as a reference architecture to evaluate the performance of our models. Note that the CNN model in our case is applied to data captured through sniffing WiFi wireless signals as opposed to wired traffic. (c) Recurrent Neural Network (RNN). The behavior of the different video traffic exhibits different patterns with respect to time. We propose using RNNs because of their superiority in training time sequence data. Specifically, we utilize the Long short-term memory model (LSTM) which addresses the vanishing gradient problem in classical RNN.

These models are presented in ***Chapter 3***.

Research Objective ②: Classifying encrypted WiFi traffic using a hierarchical classifier

Not just limited to video content, we generalize our model to more types of traffic (e.g. audio and text). Besides, there are multiple content providers for different types of traffic. For example, video content provider contains YouTube video, Netflix, and Stan. Audio content provider contains YouTube Music, Spotify, and Xiami. Web content provider contains Wikipedia, Australian Broadcasting Corporation (ABC), and Sydney Morning Herald (SMH). However, most existing approaches only classify content for one type of data. We aim to design a model that can classify data hierarchically. We present a hierarchical classifier to make coarse grained predictions (e.g. web, video, or audio) as well as fine granular predictions (e.g. content providers' platforms and exact content). We observe that traffic flows from the same type of data provider have great similarity, and hence hierarchical classifiers for coarse grain predictions can be deployed. On the other hand, in [119], the authors show that separability between different traffic flow/application categories is uneven, with some categories harder to identify. Thus, certain traffic categories would demand dedicated classifiers. We show that a hierarchical classifier is able to exploit commonality in traffic from the same data provider, but also cater for dedicated classifiers for poorly separated traffic categories. Furthermore, we show that a hierarchical approach also allows the estimation of network usage characteristics for the purposes of service management in large networks, which constitutes a first step toward identifying unknown service providers for different traffic classes.

This model is presented in ***Chapter 4***.

Research Objective ③: Generating samples by category using Bayesian non parametric based autoencoders

Deep learning classifiers are largely dependent on the quality and quantity of the dataset. Especially for the network domain, the classifier is very sensitive to the timeliness and the network environment in which the data is captured. There are many existing generative models like VAE [54] and Generative adversarial nets (GANs) [36] that can generate samples based on the data distribution. For VAE [54], the prior probability of latent space is a standard Gaussian distribution. However, using the same prior distribution of different categories makes it hard to generate data for a certain category. In some cases, we broadly collect data in a certain area, which makes we do not have label information. To circumvent this problem, in this work, we proposed two novel generators based on VAEs and Bayesian Nonparametric models (BNP), which can generate samples of a certain category without the labeled data required.

These two models are presented in ***Chapter 5***.

1.3 Research Contributions

To achieve the above three research objectives, we have proposed novel modeling methods described in Chapter 3 to Chapter 5. In each chapter, computational machine learning methods are designed and optimized based on the main research objectives. The contributions of this thesis of each work are summarized as follows (Ch3 to Ch5):

Ch3: Classifying encrypted WiFi video content using deep learning methods

In Chapter 3, (1) We demonstrate the possibility of making predictions from encrypted WiFi traffic by building deep learning based classifiers that are able to identify specific videos from a closed set of videos. (2) We show that our models are able to achieve 97% accuracy in identifying videos from a closed set of 10 videos purely based on passive measurements collected at the WiFi layer. (3) we evaluate the longevity of our classifier by making predictions two weeks apart and show that our classifier is still able to maintain the same level of accuracy.

Ch4: Classifying encrypted WiFi traffic using a hierarchical classifier

In Chapter 4, (1) We build a hierarchical traffic classifier that is able to make coarse-grained and fine-grained predictions about encrypted traffic flows by leveraging weight-sharing features in convolutional neural networks. (2) We show that our hierarchical classifier can achieve over 95% accuracy in identifying traffic types such as web, video streaming, and audio streaming as well as identifying content providers of traffic and the exact content consumed by the user. (3) We also demonstrate its potential for classifying previously unseen content to its corresponding traffic type and content provider.

Ch5: Generating samples by category using Bayesian nonparametric autoencoders

In Chapter 5, (1) We point out the importance of Bayesian nonparametric models in generating samples by categories. (2) We proposed two novel generative models based on VAE and Bayesian nonparametric models. (3) We compare our model with a state-of-art GMVAE proposed for the same generation task. We show that our models can achieve a better clustering and generation effect. (4) We explored the possibility that our models can adapt to incremental data and scale to new clusters.

1.4 Thesis Structure

This thesis mainly talks about classifying WiFi traffic using deep learning methods. The structure is shown in Figure 1.1 and introduced below:

- *Chapter 1*: This chapter introduces the backgrounds and motivations. Our research objectives, contributions, and this thesis structure are discussed in this chapter.

- *Chapter 2*: This chapter reviews the related work. The current traffic classification research progress is included in this chapter.

- *Chapter 3*: This chapter represents video streaming classification using three deep learning models. The data collection work, model structure, and experiment results are introduced in this chapter.

- *Chapter 4*: This chapter represents WiFi traffic classification, which is not only limited to video traffic but also audio and web. The chapter can seem like an extension of Chapter 3, which generalize video traffic to different types of traffic. The dataset, model architecture, and experiment results are included in this chapter.

- *Chapter 5*: This chapter introduced two novel generative models. The model definition, model architecture, and experiment results are discussed in this chapter.

- *Chapter 6*: A work summary and future research work are discussed in this chapter.

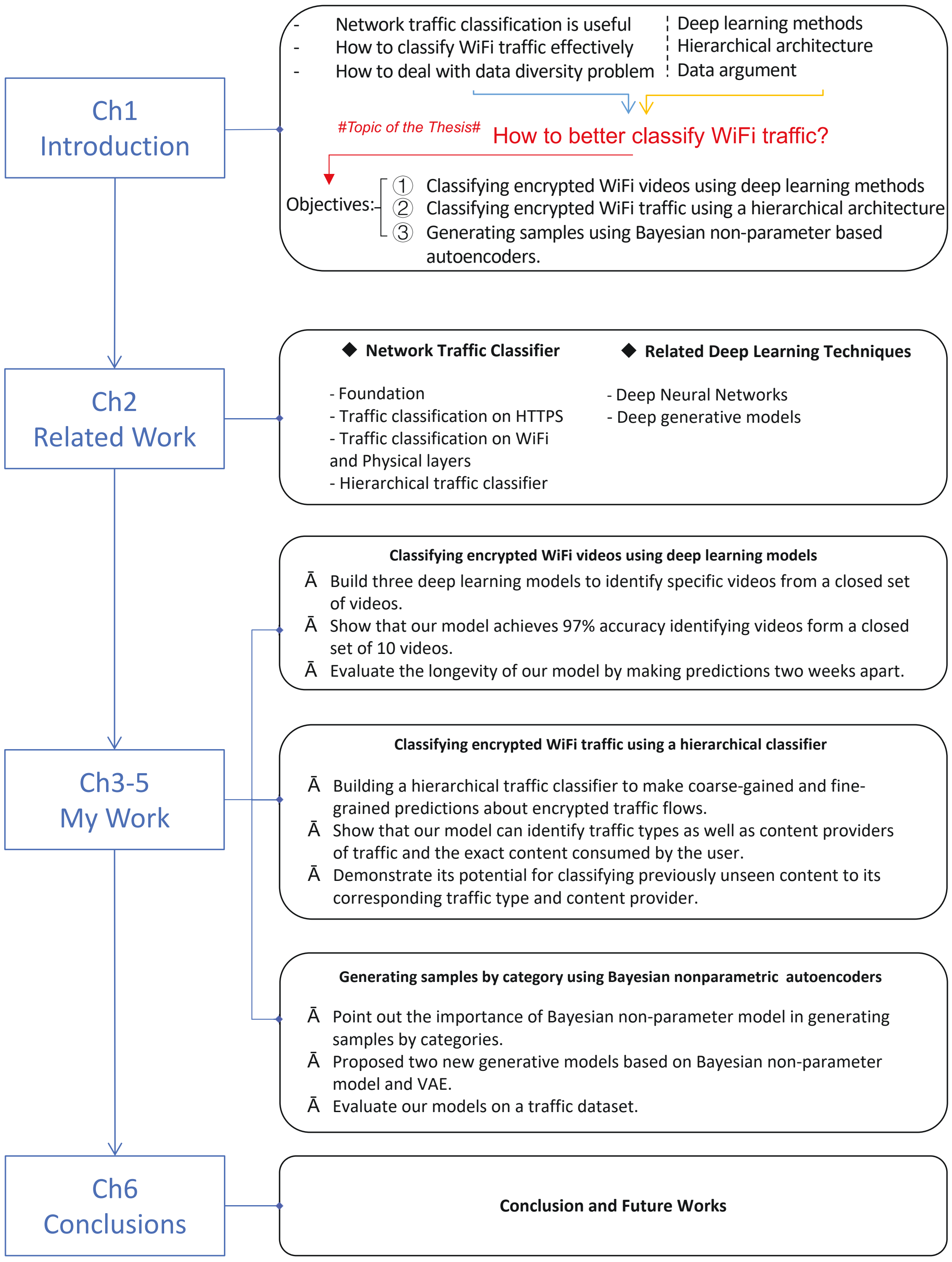

Ch1
Introduction
- Network traffic classification is useful
- How to classify WiFi traffic effectively
- How to deal with data diversity problem
Deep learning methods
Hierarchical architecture
Data argument
#Topic of the Thesis#
How to better classify WiFi traffic?
Objectives:
① Classifying encrypted WiFi videos using deep learning methods
② Classifying encrypted WiFi traffic using a hierarchical architecture
③ Generating samples using Bayesian non-parameter based autoencoders.
Ch2
Related Work
Network Traffic Classifier
- Foundation
- Traffic classification on HTTPS
- Traffic classification on WiFi and Physical layers
- Hierarchical traffic classifier
Related Deep Learning Techniques
- Deep Neural Networks
- Deep generative models
Ch3-5
My Work
Classifying encrypted WiFi videos using deep learning models
Ā Build three deep learning models to identify specific videos from a closed set of videos.
Ā Show that our model achieves 97% accuracy identifying videos form a closed set of 10 videos.
Ā Evaluate the longevity of our model by making predictions two weeks apart.
Classifying encrypted WiFi traffic using a hierarchical classifier
Ā Building a hierarchical traffic classifier to make coarse-gained and fine-grained predictions about encrypted traffic flows.
Ā Show that our model can identify traffic types as well as content providers of traffic and the exact content consumed by the user.
Ā Demonstrate its potential for classifying previously unseen content to its corresponding traffic type and content provider.
Generating samples by category using Bayesian nonparametric autoencoders
Ā Point out the importance of Bayesian non-parameter model in generating samples by categories.
Ā Proposed two new generative models based on Bayesian non-parameter model and VAE.
Ā Evaluate our models on a traffic dataset.
Ch6
Conclusions
Conclusion and Future Works

Figure 1.1 : Thesis structure

Chapter 2

Literature Review

In this chapter, we first review the traffic classification methods and techniques, aiming to give a general background knowledge of traffic classification. Then, we introduce involved machine learning techniques that are associated with the thesis works, including deep learning models, Bayesian non-parametric models, and generative models, respectively. The literature review structure is shown in Figure 2.1

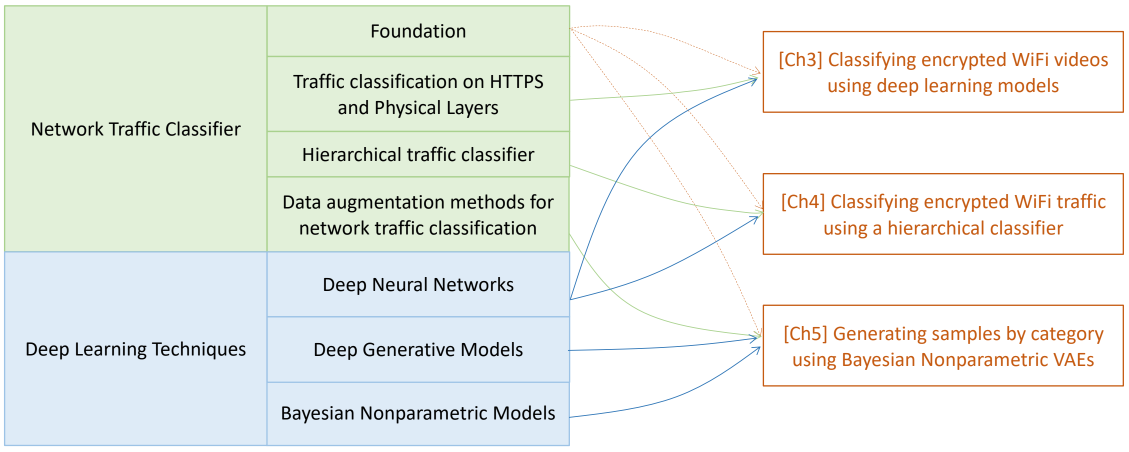

Network Traffic Classifier
Foundation
Traffic classification on HTTPS and Physical Layers
Hierarchical traffic classifier
Data augmentation methods for network traffic classification
Deep Learning Techniques
Deep Neural Networks
Deep Generative Models
Bayesian Nonparametric Models
[Ch3] Classifying encrypted WiFi videos using deep learning models
[Ch4] Classifying encrypted WiFi traffic using a hierarchical classifier
[Ch5] Generating samples by category using Bayesian Nonparametric VAEs

Figure 2.1 : Literature review section structure

2.1 Network Traffic Classifier

2.1.1 Network Traffic Classifier Foundation

With the increasing scale of the network traffic, how to effectively carry out network management and control, traffic intrusion detection, and network planning and construction is an important issue for all current Internet service providers and network operators. Traffic classification plays an important role in network management and supervision. For the existing network traffic classification methods, according to the technology used, it can be divided into port-based [1], payload-based [75; 98] and machine learning-based methods.

Port-based methods are more practical when there are limited types of application services, that is, a limited number of different types of application services can be identified according to well-known port numbers. Sen et al. [99] adopt a passive measurement method to classify the traffic of 3 popular file sharing systems (Gnutela, FastTrack, DirectConnect) based on the port-based identification. Moore et al. [77] used the toolkit CoralRef to extract the port information, and successfully realized the identification of the KaZaA stream (TCP port 1214). However, with the rapid development of peer-to-peer Internet (P2P), many applications are using dynamic port allocation. Additionally, to avoid traffic detection and filtering, various applications have adopted the port hopping technology [71], HTTP tunneling and anonymization technology [60; 108]. At the same time, the wild use of network addresses translation technology makes port-based method can only achieve about 50% accuracy [75]. Wei et al. [64]used the port-based method on three datasets respectively and found that port-based methods are easily affected by the external environment, and the classification accuracy is only 31%. Besides, more and more application traffic is tunneled through HTTPS traffic [90], which limits the performance of port-based methods [95].

Payload-based methods are traffic detection and control technology based on the application layer. It reorganizes the application layer information by deeply reading the content of the IP packet payload, to obtain the content of the application, and then classify the traffic according to the policy defined by the system. Karagianis et al. [51] studied 9 common applications and achieved traffic classification by extracting the first 16 bytes of each packet and combined with port heuristic rules. Sen et al. [98] focused on five traffic classifications (Kaza, Gnutela, eDonkey, DirectConnect, BitTorent) by fully extracting the packet payload. The experimental results showed the false positive rate and false negative rate of the five applications are less than 5%. Moore [98] analyzes the defects of the port-based method in detail, and uses 9 identification methods to construct a traffic identification system based on content characteristics. It shows that this system has a better recognition accuracy on 10 common network services. The payload-based methods have a high recognition accuracy and fine classification granularity, which has a significant effect on ensuring network management. However, with the wide adoption of end-to-end encryption, payload-based methods face huge challenges [84; 3].

Machine learning-based (ML) methods also known as statistics-based methods, have become the most popular technology in the field of traffic classification due to their advantages of wide classification range and the ability to process high-dimensional data. It have shown their effectiveness to classify encrypted traffic [101; 19; 79; 5; 6; 11; 17; 9; 40; 44; 45; 66; 69; 72; 112; 118; 61; 96; 106; 107; 114; 30; 67]. ML methods learn the patterns of different classes of network flows. In this case, three main parts of work are completed, namely characterizing traffic flows [24], modeling traffic flows [87], and extracting specific patterns [26].

Zuev et al. [125] first proposed a probabilistic model-based naive Bayes method, its recognition rate can reach 67%. However, in practice, it is difficult to guarantee the two preconditions of the naive Bayes method: the flow characteristics are independent and obey a Gaussian distribution. Based on that, Moore et al. [76] used a fast correlation-based filter algorithm [122] to extract the top 10 attribute features that are beneficial to traffic classification and used the kernel density estimation to fit the distribution function, which makes recognition accuracy is significantly improved, reaching more than 90%. In addition, Moore et al. [10] constructed a three-layer Bayesian neural network structure (246 input units, 10 output units, and several intermediate node layers), and adopted the minimum risk decision-making strategy to realize traffic recognition, and the recognition accuracy reached above 95%. However, this type of method has limitations. For example, the number of layers of network structure and the number of neurons are hard to determine and it has an over-fitting problem. Nguyen et al. and Hong et al. [78; 46] also did some research based on naive Bayes methods. ML-based methods have a good classification performance, but it is difficult to ensure that the limited training samples can fully reflect the real distribution characteristics with potential instability.

2.1.2 Traffic Classification on HTTPS Communications

Website fingerprinting and inferring user activities

Previous work explored the possibility of fingerprinting websites. Initially, these started as means of making inferences from the traffic in Tor network [82; 113] and later it was shown that the same attacks are feasible for HTTPS traffic flows [18; 83] as well. Chen et al. [18] shows that due to some basic characteristics of web applications can cause serious user information leakage. By observing HTTPS traffic, very detailed personal information can be inferred, such as a user's disease information, medications, household income, and investment secrets. For example, the authors demonstrated that when a user tries to find a specialist via the website OnlineHealth, the website search pops a drop-down list box based on the user's location requesting a very-low-entropy input (i.e limited number of doctor and specialty combinations). As a result, the attacker is able to identify the doctor, revealing the users' exact illness simply through the payload size of data. Recently, Panchenko et al. [83] used a dataset of over 120,000 web pages to show that fingerprinting websites using a collection of web pages under the same domain is not only realistic but also highly accurate. In addition, Rimmer et al. [93] proposed a deep learning-based automatic feature learning process that allows scaling the website fingerprinting to thousands of websites.

Video streaming

Schuster et al. [97] proposed a convolutional neural network to identify the exact videos from a closed set on YouTube, Amazon, Vimeo, and Netflix by observing the traffic in the network layer (e.g IP traffic observed in a router). The authors showed that packet bursts in encrypted streams correspond to DASH segment requests from the client, and that burst sizes are highly correlated with the sizes of the underlying segments. Using statistical features of the traffic flows such as down/up/all bytes per second and packets per second, the authors were able to achieve over 90% accuracy in all the platforms. In a similar study, Reed and Kranch [92] developed a system to identify the Netflix video being delivered by a TCP connection using the information provided by TCP/IP headers. The authors created a fingerprint database of 42,027 Netflix videos and showed that an accuracy of over 99.99% can be achieved. Bu et al. [15] proposed a network-in-network structure and showed that it can achieve accuracies of 98.3% and 98.5% for traffic type classification and application classification, respectively. Jin et al. [20] proposed a lightweight and online approach for large volumes of encrypted traffic classification with high accuracy and less training time. Recently Schuster et al. presented the first ever use of CNNs to detect not only the type of traffic but also the content of encrypted traffic [97]. In their proposal, the authors were able to identify which videos were downloaded over HTTPS from several video providers using features that consist of temporal representations of the traffic. The authors obtain very high accuracies within each video provider. Similar to [97], the authors of [69] proposed a CNN model to identify the class of traffic based on the characteristics of the sequence of packets in a given flow.

Instant messaging

Coull and Dyer [25] have investigated the encrypted traffic flows of the Apple iMessage service. The authors showed that it is possible for an eavesdropper to learn information about user actions, like the language of messages, the length of the messages by observing the sizes of encrypted packets and achieves 96% accuracy. Park and Kim [85] proposed a supervised learning framework to identify fine-granular user activities such as joining/leaving a chat room, adding/blocking a friend, and viewing a specific user profile in another messaging app, Kakao Talk.

VOIP and audio

Several works have investigated the side-channel information leaks in audio and voice over IP applications [116; 117; 115; 124]. Wright et al. [116] showed that due to the variable bit rate codecs used in Skype, the phrases spoken within a call can be identified by the lengths of encrypted VoIP packets. Using a hidden Markov model trained on speaker- and phrase-independent data, the authors were able to detect the presence of some phrases within encrypted VoIP calls with recall and precision exceeding 90% simply by leveraging the packet lengths of the UDP based RTP protocol. Zhu and Fu [124] extended the work and showed that the distribution packet lengths can also be used to identify exact speakers in a VOIP call.

2.1.3 Traffic Classification in WiFi and Physical Layers

One of the non-trivial aspects of side-channel attacks is the possibility of making inferences through WiFi sniffing. By the time packets are sniffed at the WiFi layer, the packets are already encrypted twice. Several works described above highlighted that the traffic artifacts of HTTPS flows are still visible in WiFi frames allowing the possibility of fingerprinting content consumed by the user [65; 18]. In addition to inferring direct activities of the users whilst online, Li et al. [63] showed that long-term observation of encrypted traffic flow also reveals demographic information such as gender and education level.

Recent research has successfully identified video streaming content in encrypted traffic without using deep learning techniques [91; 41]. In particular [91] focused on identifying video streaming using a Variable Bit Rate algorithm within encrypted WiFi traffic using similarity metrics and statistical machine learning. In a similar manner, but with the addition of DASH streaming, [41] identified video streaming within WiFi traffic. In their latest work [41], the authors adopted an approach similar to [91] by modeling various video streaming traffic. They also applied multiple statistical machine learning techniques but did not propose a generic method applicable to deep learning techniques. Overall, both methods obtained similar accuracies of 90%.

More recently, O'Shea et al. [80] further extended the side-channel attacks to the physical layer and showed that the burstiness created in the application layer is still visible in the physical layer and can be leveraged to make inferences about the actual communication taking place over the radio interface. Using the IQ symbols in the physical layer, the authors trained an LSTM recurrent neural network that is able to identify some fine granular traffic types such as YouTube, Spotify, and GitHub activities. Similar physical layer fingerprints were also found in LTE networks [56].

2.1.4 Hierarchical Traffic Classifier

A limited amount of work has looked into the possibility of building a generic traffic classifier that can handle different classes of traffic such as web, video streaming, and audio streaming. For instance, Grimaudo et al. [39] proposed a hierarchical classifier that classifies IP traffic into over twenty fine grain classes with four high-level classes. However, in this work, the "unknown" classes are identified at the top of the hierarchy and as a result, the classifier can not relate the unknown traffic types to much closer existing classes. Also, the methodology is based on a "hierarchy of classifiers" rather than a single unified classifier. Yu et al. [121] proposed a hierarchical multi-class SVM classifier mostly focusing on various types of P2P traffic. Moreover, it has to be noted that both of these works operated at the IP traffic level where flow information is available.

2.1.5 Data Augmentation Methods for Network Traffic Classification

Deep learning (DL) methods have been increasingly applied to solve encrypted traffic classification problems. However, the performance of a classifier is largely dependent on the quality and quantity of datasets. And, it is very sensitive to the timeliness and the network environment in which the data is captured. In [49], the authors show how model performance is affected when the testing and training datasets are collected at different geographical locations.

Data imbalance is another problem encountered by most DL methods[35]. Most specifically, in the network domain, some protocols/applications over other types of traffic[43]. Data balancing was proposed to overcome biases encountered when training models with imbalanced data [22]. Yu et al. [42] proposed a GAN-based Traffic Augmentation (TA-GAN) for imbalanced traffic classification. It is an end-to-end framework that integrates the generation of the minority traffic samples with the training of the target classifier. In addition, Hasibi et al. [43] proposed a novel traffic data generation method using LSTM networks to generate traffic flows and estimate kernel density for replicating the numerical features of each class. Most recently, Pan et al. [111] proposed a novel traffic data generation approach named PacketCGAN based on the use of conditional GAN, which can control the modes of data to be generated. PacketCGAN uses Conditional GAN to generate specified samples with the input of applications' types as conditional and thereby achieve data balancing. There is still a lot of research space for data argument. More especially, in the network domain, the collection of labeled Internet traffic requires a specific setup in a private network (e.g., lab), to capture the traffic and filter it based on specific parameters [95]. In this case, how to generate samples without labeled data is an important issue.

2.2 Related Deep Learning Techniques

In this section, we review some deep learning models in detail that are used in the following chapters. Since the remaining chapters focus more on the corresponding models for the specific thesis objects, we introduce them here.

2.2.1 Deep Neural Networks

Deep neural networks consist of multi-layer perceptrons. It stimulates the internal operations of neurons through functions including parameters and weights, simulates the connections between neurons with the superposition of nonlinear operations, finally realizes the reintegration of information, and then outputs the results of classification or prediction. For the difference between the output result of the neural network and the real result, the neural network will adjust the corresponding weight layer by layer through the gradient to narrow the difference, to achieve the best performance.

MLP is a neural network model, which can solve the linear inseparable problem that cannot be solved by a single-layer perceptron. The neurons in the input layer receive the input signal, and each neuron in the hidden layer and the output layer is fully connected. When the multilayer perceptron is used for classification, the number of input neurons is the dimension of the input data, the number of output neurons is the number of categories, and the number of hidden layers and hidden layer neurons depends on the specific situation. MLP is a basic architecture in deep learning models and is still used in many applications.

CNN is wildly used in image-based applications. Neocognitron [31] can be considered the first implementation of CNN, although there is no global supervised learning process like the backpropagation algorithm in Neocognitron. The convolution and downsampling are inspired by the Hubel-Wiesel concept [48]. Based on this work, LeCun et al. used a backpropagation algorithm to design a CNN, which is known as LeNet [58]. LeNet is a classic CNN structure, and many subsequent works have been improved on this basis. The basic structure of a CNN model consists of an input layer, multiple convolutional layers, multiple pooling layers, multiple fully connected layers, and an output layer. The convolutional layer consists of multiple feature maps, and each feature map consists of multiple neurons, each of which is connected to the local area of the feature map in the previous layer through a convolution kernel. The convolution kernel is a weight matrix (e.g. $3*3$ or $5*5$ for 2D) [58]. The convolutional layer of CNN extracts different features of the input through the convolution operation. The pooling layer follows the convolutional layer. The Pooling layer chooses a certain way to reduce the dimension of the feature map(e.g. max pooling, average pooling). It can not only reduce the amount of computation but also effectively avoid overfitting. After multiple convolutional and pooling layers, one or more fully connected layers are connected. Similar to MLP, each neuron in the fully connected layer is fully connected to all neurons in the previous layer. Fully connected layers can integrate class-discriminative information in convolutional and pooling layers [94]. Compared with MLP, the weight sharing of convolutional layers in CNN reduces the complexity of the network model and reduces overfitting, and thus obtains a better generalization ability [47].

RNN [123] is a type of neural network model for processing and predicting sequence data. The neural network with recurrent structure overcomes many limitations of traditional machine learning methods. Traditional Neural networks can only process one input at a time, that is, the previous input and the next input are unrelated. However, some tasks require handling sequences, that is, the previous input is related to the following input. For example, when we understand the meaning of a sentence, it is not enough to understand each word of the sentence in isolation. RNN and its variant networks have been successfully applied to a variety of tasks, especially when there is a certain time dependency in the data. However, RNNs are usually difficult to train. After many loops, the gradient tends to disappear. Aiming at this problem, LSTM [102] was proposed, which can maintain long-term memory of information and has better performance for processing long sequences.

2.2.2 Deep Generative Models

Unsupervised generative models

VAE is a deep generative model based on the structure of the auto-encoder [54]. The encoder maps the samples to the latent variables in a low-dimensional space, and then restores the latent variables to reconstructed samples through the decoding process. It assumes that the prior of latent variables is a normal distribution. This assumption is based on computation convenience, without considering rationality, which will inevitably affect the generation ability of the model. Importance weighted autoencoders [16] is one of the most important improvement methods for VAE models. From the perspective of the variational lower bound, it improves the performance of the generative model by weakening the role of the encoder in the variational lower bound.

Many deep generative models generate high-resolution image samples combined with CNN. Deep convolutional inverse graphics network [57] integrates convolutional layers into VAE. The structure of the deep convolutional inverse graphics network is the same as VAE. It replaces the encoder and decoder in the VAE from the original fully connected layers to the convolutional and deconvolutional layers. Adversarial autoencoders(AAE) [73] is a generative model that applies the idea of adversarial to the VAE training process. The main difference between these two is that VAE uses the KL divergence of the prior distribution and posterior distribution to constrain the hidden variables, while AAE constructs a pseudo-prior distribution to match the real posterior distribution, which attaches an adversarial network at the hidden variable layer. This adversarial network consists of a generator and a discriminator. The generator of AAE uses the same neural network as the encoder to fake the distribution of the hidden variables close to the real, and the discriminator is responsible for distinguishing the samples obtained from the true and false distributions.

Generative adversarial nets (GANs) [36] is commonly used in the field of image generation. It converts the difficult-to-solve likelihood function into a neural network, allowing the model to train the appropriate parameters to fit the likelihood function. It consists of a generator and a discriminator. The goal of the generator is to generate realistic fake samples so that the discriminator cannot distinguish between true and false. The goal of the discriminator is to correctly distinguish whether the data is a real sample or a fake sample from the generator. In this process, these two components need to continuously optimize their generating ability and discriminative ability, and the result is to find the Nash equilibrium between these two. When the discriminator cannot make a correct judgment, the generator learns the true data distribution. The discriminator and generator use convolutional and deconvolutional networks, and each layer uses batch normalization.

Wasserstein GAN(WGAN) [8] theoretically analyzed the defects of the original GAN, and proposed to replace the KL divergence and JS divergence with Wasserstein distance. This change improved the stability of the model. Deep convolutional generative adversarial networks (DCGAN) [88] is an important improvement of GAN. It selects the best generator and discriminator architecture from various structures so that the stability has been significantly improved. The deconvolution structure in the generator has a limited size of the mapping area on the image, which makes it difficult for DCGAN to generate high-resolution images. The new-generation GANs, such as BigGAN [14], use residual networks to highly improve the generative ability. It can handle image details very well and generate very realistic 512×512 natural scene images. But, a large scaled training set is required in the training process.

Supervised and semi-supervised generative models

Now, let's review generative models that can generate samples by category. Auxiliary deep generative models(ADGM) [70] represent label information with one-hot vectors, which enables VAE to process supervised data, essentially adding a conditional constraint to the encoder. This model adds label factors when learning samples so that VAE can generate corresponding types of samples according to the specified labels. GMVAE [27] replaces one single normal distribution with a Gaussian mixture as the latent variables' prior distribution. Their target is unsupervised clustering learning, while the number of categories is required.

Mirza and Osindero [74] proposed a novel way to train GAN, which feeds label data y to determine which category data be generated. The original GAN generated images randomly. Therefore, the authors consider adding some conditional information, such as category labels, so that the image generation can proceed in the specified direction. Specifically, Conditional GAN adds conditional information y to the input of the generator and the discriminator, and only the images generated by the generator can pass the discriminator only if they are real enough and match the conditions. Zhiyue et al. [68] proposed a category-aware GAN(CatGAN). It measures the distance between real samples and generated samples in each category and then guides the model to generate high-quality category samples by reducing this distance.

The above methods focus on data generation by category. And the label information is needed in the training process.

2.2.3 Bayesian Nonparametric Models

In traditional mixture modeling approach models, the number of parameters in the model is fixed and does not change with the change of the data. However, the BNP methods determine the complexity of the model according to the observed data. Furthermore, BNP models allow for increased complexity as more data are observed. For example, for unsupervised clustering tasks, BNP methods can learn the cluster number automatically from the data itself. Compared to parameterized models (e.g. K-means, Gaussian mixture mode), it is much better to set the number of clusters empirically. Because of the nonparametric nature of their prior distribution, they have a strong ability to describe the data [32]. This is an important advantage of non-parametric models. Analyzing data using BNP models generally follows the pattern of Bayesian data analysis. Each model describes the data generation process that contains hidden variables. This procedure clarifies the statistical assumptions made by the model and specifies the joint probability distribution of hidden and observed variables. Given an observed dataset, data analysis is performed by posterior inference, computing the conditional distribution of the hidden variables given the observed data. posterior inference finds the distribution of hidden structures that might generate the observed data. The difference between BNP and other Bayesian models is the assumption that the hidden structure grows with the data. Its complexity, such as the number of mixture components or the number of factors, is a part of the posterior distribution. It does not need to be specified in advance but is determined as part of the data analysis.

The following will briefly introduce the models and inference methods that we used in the thesis.

Dirichlet Process

Dirichlet process (DP) is a stochastic process defined in probability measure proposed by Ferguson in 1975 [29], and its parameters are centralized parameter $\alpha > 0$ and base probability distribution G. It is denoted as $G - DP(a, G)$. The probability distribution obtained by the Dirichlet process is discrete and therefore it is very suitable for constructing mixture models. For example, Antoniak et. [7] constructed a Dirichlet process mixture in 1974 by adding a generative probability to each data point.

$$x_i \sim p(x|\theta_i) \tag{2.1}$$

where $\theta_i \sim G$ are the parameters of the probability distribution generated data point.

Stick-breaking construction [100] is a constructive formulation of the Dirichlet process. Specifically, a stick of unit length is cut, and each cut is in proportion to the random variable of the beta distribution:

$$v_k = Beta(1, \alpha), \pi_k = v_k \sum_{j=1}^{k-1} (1 - v_j) \tag{2.2}$$

As shown in Figure 2.2, for a 1 unit length stick, a v_1 length is cut for the first time, and the v_k proportional length of the remaining part is cut each time after that. The Dirichlet process of cutting sticks is the basis of variational inference [12].

Variational Inference

The variational inference method is a widely used approximate optimization method [105]. Many problems have been solved in the fields of physics, statistics, financial analysis, and control science. In the field of machine learning, variational inference methods also have many applications. Through variational analysis, non-

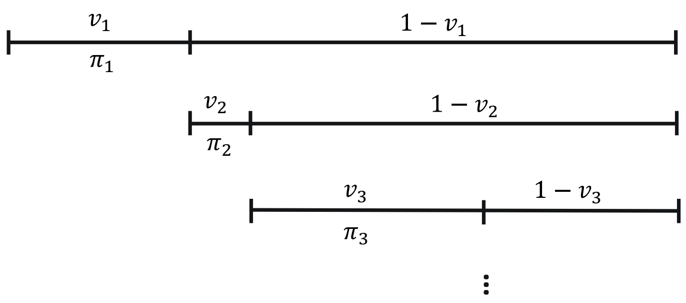


Figure 2.2 : Illustration of stick-breaking construction

optimization problems can be transformed into optimization problems. Some difficult problems can also be solved by the variational approximate method [110].

In the variational Bayesian method, given the dataset D and the posterior distribution $p(\theta|D)$. The variational method defines the approximate distribution of the posterior distribution as $q(\theta)$. Using Jason's inequality, evidence lower bound (ELOB) of the log-likelihood can be obtained.

$$\log p(D) \geq E_q[\log p(\theta, D)] - E_q[\log q(\theta)] \tag{2.3}$$

By maximizing the ELOB, the optimization process can be completed. Therefore, the idea of variational inference is to transform the original problem into an optimization problem to solve the approximate distribution $q(\theta)$ and combine an effective optimization algorithm to complete the task of Bayesian inference [13; 13]. There are some parameters θ and hidden variables h in the model. The optimization can be solved by the variational expectation maximization method: by introducing the mean-field assumption$q(\theta, h) = q(\theta)q(h)$. And then the variational expectation-maximization algorithm can be performed iteratively [81].

Chapter 3

Classifying Encrypted WiFi Videos Using Deep Learning Models

This chapter aims to achieve our *objective*①: Classifying encrypted WiFi videos using deep learning models.

3.1 Introduction

As introduced in Chapter 1, more than 95% of global Internet traffic is currently end-to-end encrypted [62; 38]. It makes there a strong need to make inferences and predictions from encrypted traffic flows. Previous research explored the possibility of making inferences based on encrypted traffic flows by capturing network data packets at the IP layer [97; 4; 25]. This is viable because despite the message content being encrypted, the statistical properties of traffic flow such as packet lengths, inter-packet times, burst sizes, and burst intervals can still reveal information about the underlying encrypted traffic that is in transit. Also, at the IP level, there are other useful meta-data such as IP addresses, ports, and TLS header information.

This work investigates the possibility of making useful inferences from passively observed WiFi traffic that is encrypted at both transport layer and MAC layer. This is more challenging in comparison to making predictions from the IP layer traffic due to the lack of any meta information.

Videos are highly popular on the Internet but are frequently misused in many ways. Therefore, we focus on identifying traffic flows from a set of known online videos. To summarize this chapter's work, we list the main contributions as follows:

- We demonstrate the possibility of making predictions from encrypted WiFi traffic by building deep learning-based classifiers that are able to identify specific videos from a closed set of videos when they are streamed from a popular video service, i.e. YouTube via Dynamic Adaptive Streaming over HTTP (DASH).
- We show that a simple Multi-Layer Perception architecture is able to achieve 97% accuracy in identifying videos from a closed set of 10 videos purely based on passive measurements collected at the WiFi layer.
- Finally, we evaluate the longevity of our classifier by making predictions two weeks apart and show that our classifier is still able to maintain the same level of accuracy.

3.2 Method

3.2.1 DASH Streaming

Video streaming over the Internet has shifted to what is commonly referred as HTTP-based Adaptive Streaming (HAS). HAS works by encoding multiple versions of original content with different bit rates and resolutions. The encoding bit rates recommended by YouTube for video streaming are 8, 5, and 1Mbps for video resolutions of 1080p, 720p, and 360p respectively for frame rates up to 30fps. Each version of the video is split into smaller chunks (or segments), typically 2-4 seconds in length. These chunks are then stored on a web server which can be accessed by clients on demand using simple HTTP GET requests from a video streaming client running one of the HAS algorithms. Since HTTP operates on top of TLS, all data including HTTP headers are encrypted and can only be decrypted at the endpoints. For each video, the relationships between the chunks and video information are stored in a manifest file. A user trying to stream a particular video is

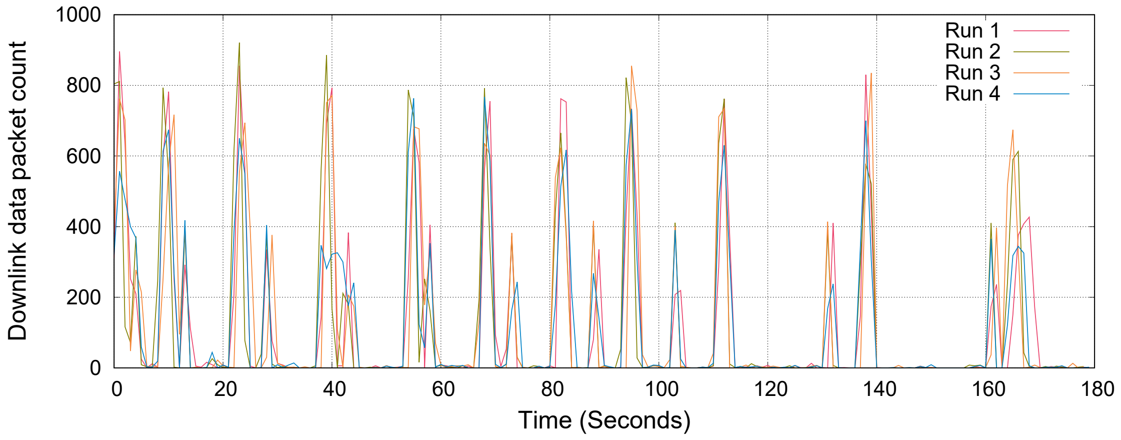


Figure 3.1 : I/O graphs of different traffic flows for the same video

first presented with this manifest file. The client video player (known as the DASH player) uses information from a server manifest file and current network conditions to dynamically adapt the stream.

As a result, streaming video with the DASH player creates a traffic profile that typically contains periodic spikes of downloads of potentially different magnitude based on the network condition (illustrated in Figure 3.1). These spikes are related to downloading the next series of chunks of the video followed by a waiting time until the user has played a certain percentage of the downloaded chunk. In particular, we can see in Figure 3.1 how the DASH player adapts to various network conditions by requesting different quality chunks, e.g. “Run 4” contains fewer packets compared to other runs.

3.2.2 Preprocessing & Feature Engineering

Data Filter

The WiFi captures include any data packet transmitted on our selected channel within proximity of the air environment. These packets were encrypted by IEEE 802.11 protocol using WiFi protected access 2 (WPA2), therefore it was impossible to extract any layer 3 and above protocol information such as port numbers or to

Table 3.1 : Feature selection from wireless traffic data

Traffic Direction	**Feature Name**
Uplink	F1. Number of Packets (data)
Downlink	F2. Number of Bytes (data)
Combination (up and down)	F3. Number of Packets (non-data)
	F4. Number of Bytes (non-data)
	F5. Minimum packet size
	F6. Maximum packet size
	F7. Average packet size
	F8. Variance packet size

apply deep packet inspection. Instead, we obtain several basic parameters from the MAC layer, for instance, the frame size, frame type, frame duration time, radio information including signal strength and noise level, and MAC addresses of the source and destination.

Frame Type Identification

According to the IEEE 802.11 protocol, the MAC frames of the filtered target data are divided into three types: management frames, control frames, and data frames, of which data frames are most closely related to video classification. Hence, data packets are selected from the target laptop using the packet size parameter to only capture frames with a size greater than or equal to the minimum packet size in a wired network.

Feature Engineering

The captured data consists of the first three minutes of each video stream. This traffic is later grouped into up-link, down-link, and combination (up and down) frames. For each group, we binned each feature in 500 bins of 0.36 seconds for ease of statistical computations. Features are generated from a sliding window over these bins.

In each temporal bin, we compute the features that characterize the dynamic aspect of the traffic: number of packets in data frames, number of bytes in data frames, number of packets in management and control frames, and number of bytes in management and control frames. In addition, the traffic waveform of MPEG-DASH video streaming from YouTube fluctuates depending on the video content. Accordingly, four additional features, namely the minimum size of packets, maximum size of packets, average size of packets, and variance of packet size, all within the studied time period, are constructed to further characterize video streaming traffic. These features are summarised in Table 3.1.

3.2.3 Classifier Architectures

Convolutional Neural Network Model

As explained in Section 2, Schuster et al. [97] apply a CNN model to classify the content of the traffic. This is similar to the goal of our work. The CNN model used in [97] reported excellent performance, and we implement this model as a reference architecture to evaluate the performance of our models. Note that the CNN model in our case is applied to data captured through sniffing WiFi wireless signals as opposed to wired traffic. The architecture of the CNN model in Figure 3.3(a) is cascaded by an input layer, three convolution layers, a max pooling layer, and two fully connected layers.

Long Short-Term Memory Model

The behavior of the different video traffic exhibits different patterns with respect to time as illustrated in Figure 3.2.

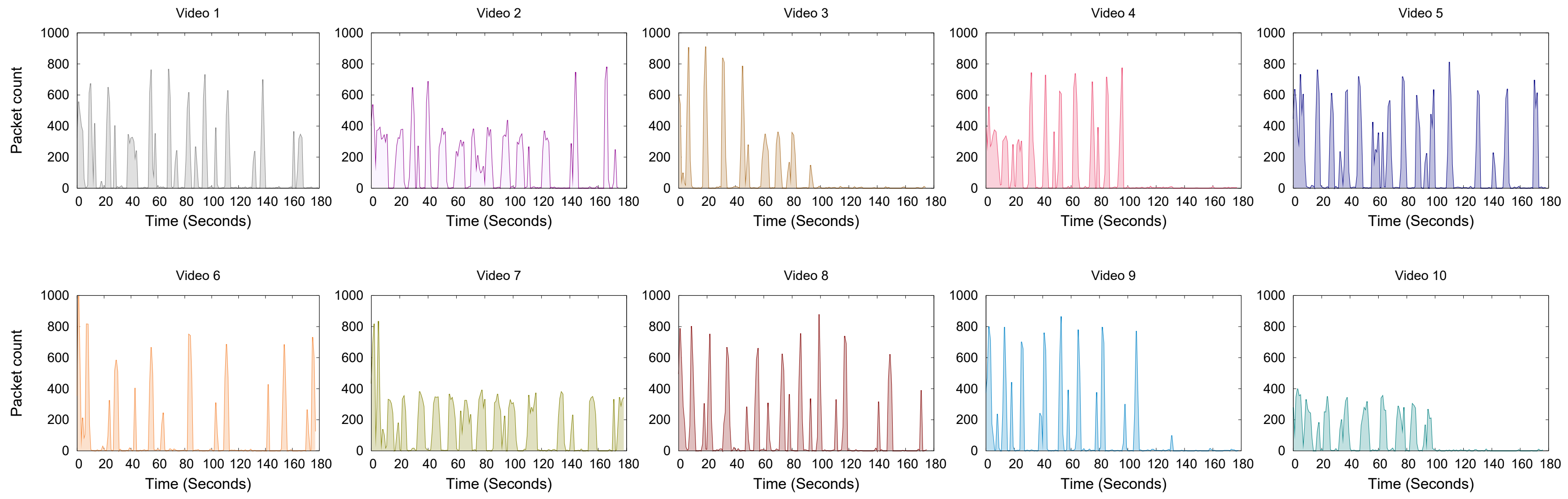


Figure 3.2 : I/O graphs of a single run for 10 different videos

These graphs are exported from WireShark's statistic tools, where the X-axis is the time sequence with 1s interval and the Y-axis is the count of data packets on the down-link flow. The distribution of the spikes in different video traffic follows a specific time sequence. This indicates that the time correlation of the feature values is crucial. Hence, we propose using RNNs because of their superiority in training time sequence data. Specifically, we utilize LSTM models which address the vanishing gradient problem in classical RNNs.

As shown in Figure 3.3(b), the input to the LSTM network is a fixed-size 500×1 array in which 500 denotes time steps and 1 denotes a single feature. This network has two hidden layers and each hidden layer has 32 neural nodes.

Multi-Layer Perceptron Model

During training, if just one feature was chosen the input to our network is a fixed-size 500 array. The array is passed through a stack of Fully-Connected layers. Through experiments, we acquired a high classification accuracy using two hidden layers and one dropout layer. The first hidden layer has 300 neural nodes and the second hidden layer has 100 neural nodes as shown in Figure 3.3(c).

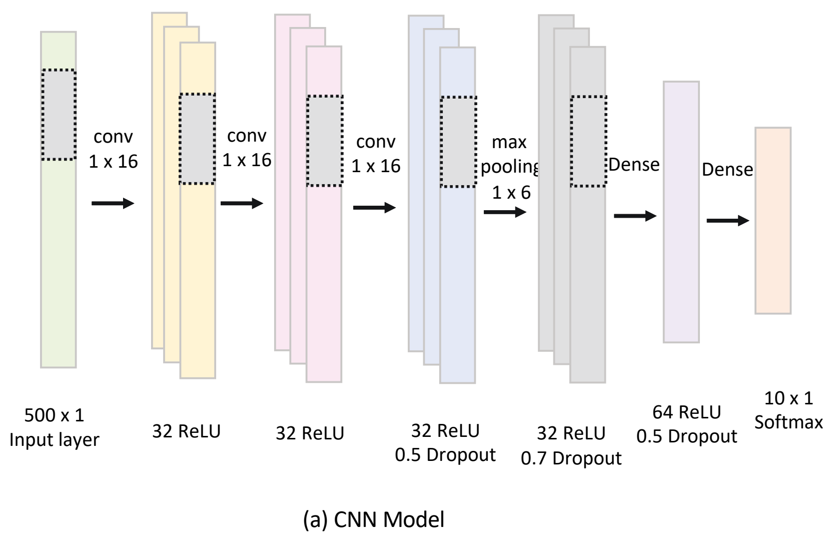
3.3 Experiments and Results

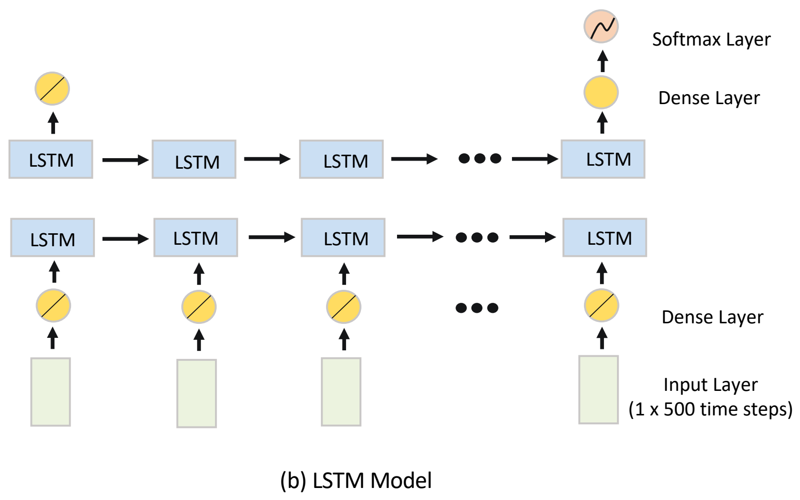
3.3.1 Dataset and Evaluation Metric

To explore the feasibility of eavesdropping attacks over an encrypted wireless network, we configured a laptop to connect to an 802.11n WiFi access point using channel 6 of the 2.4 GHz spectrum with WAP2 encryption. From this laptop, we repeatedly downloaded the same 10 videos from YouTube.[*] On a separate laptop, we used AirPcap Nx from Riverbed[†] to passively capture all the frames available

[*]List of YouTube videos is presented on https://cloudstor.aarnet.edu.au/plus/s/kZekE1nN3r0Xvog

[†]`https://www.riverbed.com`





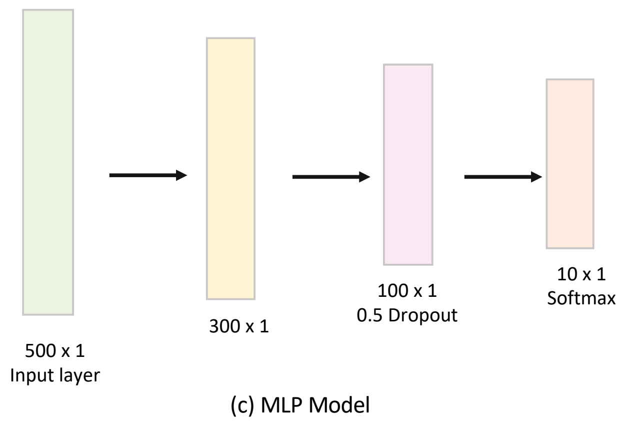


Figure 3.3 : Architecture of different models

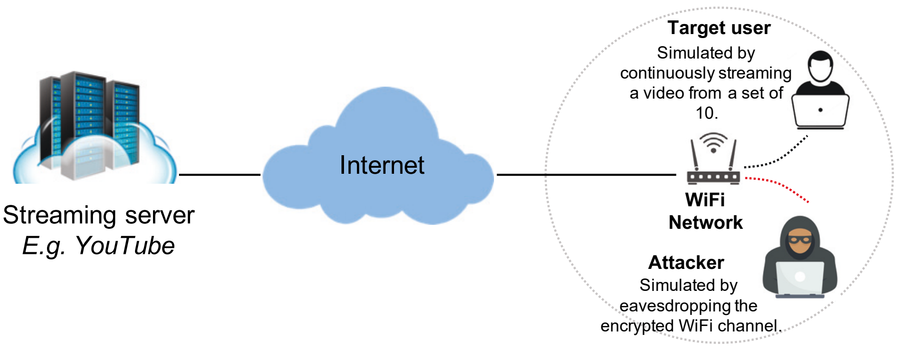

Target user
Simulated by continuously streaming a video from a set of 10.
Internet
WiFi Network
Streaming server
E.g. YouTube
Attacker
Simulated by eavesdropping the encrypted WiFi channel.

Figure 3.4 : Data collection setup

on this channel regardless of the Ethernet address within the frame. This setup is illustrated in Figure 3.4.

Overall, we captured wireless traffic in a campus environment of 10 videos for more than 300 times for each video. For each video, we only captured the first three minutes of the stream. In addition, a registered YouTube Red account was used to avoid advertisements. These files were later post-processed using the Scapy Python library to extract frames associated with the targeted laptop Ethernet address. We called this dataset as YouTube traffic dataset. We shuffle the captured traffic and then split them into training and testing sets in ratios of 80% and 20% respectively.

We adapt classification Accuracy for evaluation. The accuracy represents the proportion of correctly predicted samples in all samples. It is the most intuitive evaluation metric in classification problems. As our dataset is balanced, accuracy is a good measure of our models.

3.3.2 Implemention Details

We implement our models with Tensorflow [2]. The batch size is set as 64. We use Adam optimizer [53] and set learning rate as 0.0001. The activation function is ReLU [34]. The batch normalization decay and batch normalization epsilon are set as 0.5 and 0.001. The total training epoch number is 100.

Table 3.2 : Feature evaluation for CNN model

Feature Name	**Down-link**	**Up-link**	**Sum**
Packet number (data)	0.96714	0.97183	0.96870
Packet size (data)	0.96575	0.96714	0.96870
Packet number (non-data)	0.96714	0.97500	0.96870
Packet size(non-data)	0.96714	0.97183	0.96870
Minimum packet size	0.96244	0.94992	0.93740
Maximum packet size	0.96557	0.95305	0.96714
Average packet size	0.97027	0.94992	0.96400
Variance packet size	0.94053	0.91862	0.96557

3.3.3 Performance

In this section, we evaluate and demonstrate the effects of using our selected features in combination with the three neural network models outlined above.

Classification Performance on CNN Model

As previously mentioned, during the data collection period about 3,198 video streaming samples of 10 videos were captured from YouTube Red. The input of the CNN model is a 500×1 array where 500 denotes the number of temporal bins of 0.36 seconds and 1 denotes one feature. Each feature is utilized one by one to train the model and the corresponding test accuracy results are presented in Table 3.2.

The performance of the CNN model, as shown in Figure 3.5 and Figure 3.6(a), is on par with results from Schuster et al. [97]. As mentioned in Chapter 1, our data in this thesis was captured by WireShark through AirPcap, not directly through the WLAN interface. Thus, there is much more noise in our captured data compared to the wired data from [97]. However, the accuracy of the trained model appears to be robust against such noise.

Table 3.3 : Feature evaluation for LSTM model

Feature Name	**Down-link**	**Up-link**	**Sum**
Packet number (data)	0.93427	0.95931	0.93114
Packet size (data)	0.90923	0.90141	0.94366
Packet number (non-data)	0.93271	0.96088	0.93271
Packet size(non-data)	0.90767	0.93897	0.83881
Minimum packet size	0.94679	0.86385	0.87793
Maximum packet size	0.79186	0.90454	0.90767
Average packet size	0.78247	0.89202	0.95775
Variance packet size	0.83099	0.71831	0.85916

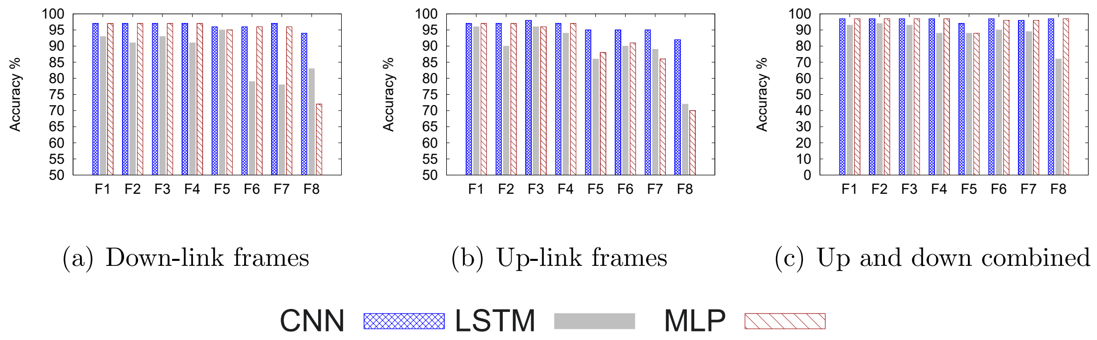
Classification Performance on LSTM Model

We configured the LSTM model to select a single feature at a time. However, the input to the LSTM model is different from that of the CNN model because it is a sequence of 500 steps corresponding to the number of temporal bins of 0.36 seconds from the traffic traces. The corresponding test accuracy results are presented in Table 3.3.

Similarly, as with the CNN model, we applied each feature to train the LSTM model. As we can see in Figure 3.5, the LSTM model performs relatively well to detect and classify the video with an accuracy ranging from 72% for packet size variance to 96% for the number of packets in the sliding window in the uplink. In particular, we see that the LSTM model performs slightly worse than the state-of-the-art CNN model [97]. To better understand these results we present in Figure 3.6(b) the confusion matrix of the LSTM model.

Classification Performance on MLP Model

Here, a video streaming flow is represented as a 500×1 array input to the MLP model with only one feature chosen. To seek the optimal MLP configuration scheme, hyper-parameters such as feature selection, the number of hidden layers, the number of nodes in each hidden layer, and the choice of activation function were evaluated



(a) Down-link frames (b) Up-link frames (c) Up and down combined

Figure 3.5 : Accuracy of various neural network models

Table 3.4 : Feature evaluation for MLP model

Feature Name	Down-link	Up-link	Sum
Packet number (data)	0.97340	0.96714	0.97183
Packet size (data)	0.96557	0.97027	0.97340
Packet number (non-data)	0.97496	0.96401	0.97496
Packet size(non-data)	0.97183	0.96714	0.96714
Minimum packet size	0.95149	0.88420	0.87950
Maximum packet size	0.96244	0.91393	0.96244
Average packet size	0.96244	0.86072	0.96714
Variance packet size	0.71830	0.70423	0.96870

by both empirical analyses and experimental results. The test accuracy results are presented in Table 3.4.

The results demonstrate that the MLP structure with 500×1 input (i.e. only one feature was selected) and 2 hidden layers could provide excellent performance (shown in Figure 3.5).

We observe from Figure 3.5 that, surprisingly, feature F3 "Number of Packets (non-data)" achieved the best performance not only on down-link traffic but also on bi-direction traffic. Examining up-link traffic, feature F2 "Number of Bytes (data)" performed best. Combining the traffic direction and feature type, the "Number of Packets (non-data)" on down-link traffic of video streaming trained the most accurate model (accuracy of 97.5%). Note that increasing the number of hidden

Table 3.5 : MLP model accuracy with new dataset two weeks later

Feature Name	**Down-link**	**Up-link**	**Combined**
Number of Packets (data)	0.95667	0.96334	0.97334
Number of Bytes (data)	0.95667	0.96334	0.97000
Number of Packets (non-data)	0.95334	0.96334	0.96000
Number of Bytes (non-data)	0.97000	0.95667	0.96000
Minimum packet size	0.95000	0.85000	0.86334
Maximum packet size	0.95000	0.89667	0.95667
Average packet size	0.97000	0.81667	0.97000
Variance packet size	0.67334	0.67667	0.97334

layers and changing the position of the dropout layer did not achieve any further gain in performance.

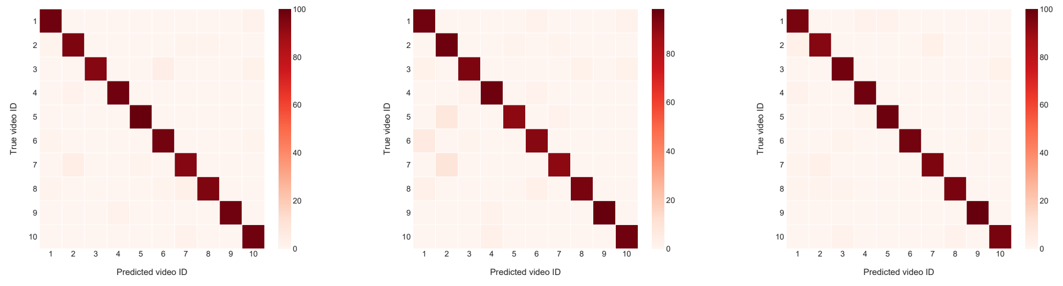
To better understand the performance of our MLP classifier, we present in Figure 3.6(c) the confusion matrix for all ten videos based on the optimum model.

Other feature combinations were also validated using an MLP model with an input vector of $500 \times k$ entries where k denotes the number of features. The input vector was constructed by splicing the 500×1 vectors of different features together. The performance was lower than those obtained using a single feature.

To demonstrate time independence of the MLP model, we captured the traffic flows for the same 10 videos after 2 weeks and tested it again on the MLP model. The results are shown in Table 3.5. It is clear that the performance of the model on new test sets and old test sets is maintained.

3.3.4 Performance Analysis

In this thesis, three neural network architectures were implemented and used to train a video classification model. The results aforementioned showed that all models achieve similar performance. Next, we analyze the three deep learning architectures in detail and clarify the limitations of our models.



(a) CNN (b) LSTM (c) MLP

Figure 3.6 : Classification performance confusion matrix for models with F1 (number of packets (data) on down-link)

Firstly, we focus on the optimal model structure to explore the underlying factors for acquiring high MLP performance. In the MLP architecture, each of F1, F2, F3, and F4 which are directly obtained from the captured traffic files could achieve excellent performance on down-link, up-link, and combination of bi-directional links. Intuitively, it is expected that the number of packets in the data frame provides the best performance because the video contents are encapsulated in data frames. This result can be explained by the number of packets per second of different videos as shown in Figure 3.2. This figure validates that the fluctuation of "number of packets" along with time is a crucial characteristic to distinguish between videos. This was also demonstrated in [97] for the case of a wired network capture. On the other hand, it is worth noting that non-data frames can also be exploited to classify video streaming accurately. This demonstrates that not only the bursts generated by video content but also the interaction information between server and client carry information.

In addition, the performance of other features (F5, F6, F7, and F8) generated from the originally captured data is not stable. From Figure 3.1, it is clear that the traffic waveforms of the same video captured at different times share many common characteristics, especially the time sequence of wave peak and wave trough. However, the amplitude variation of these waves is dependent on the WiFi signal environment

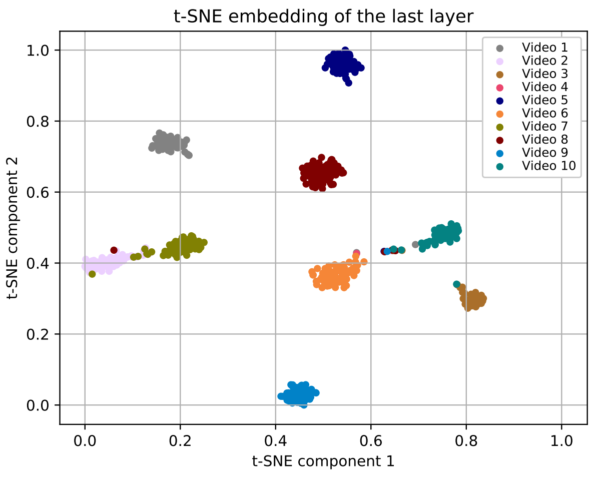


Figure 3.7 : T-SNE embedding of the last layer

due to the DASH rate adaptation. Therefore, the probable reason that the generated features could not provide steady or reliable performance is that the noise would be magnified by the generated features. This would result in overfitting during the model training.

Next, we analyze the performance of both CNN and LSTM models. CNNs are advantageous in the image processing field [59]. However, seeking suitable and related features to generate meaningful convolution in other application domains is a challenge, in spite of prior work [97]. It is worth noting that the paper [97] stated significant noise in traffic as a limitation of their model. As we sniff the traffic signal using AirPcap, the same drawback applies. Moving to the LSTM model, LSTM schemes can classify dynamic time sequence behavior and can process arbitrary input series of time sequence using its internal memory. Video streaming is by definition a time sequence, hence our consideration of LSTM. Even though we do not use video content but focus on fingerprints extracted from the original video, the performance of the LSTM model is considered satisfactory.

In order to better visualize the optimal MLP classification, we apply t-SNE on the output of the last hidden layer of the original dataset and present the result in

Figure 3.7. In this figure, we can see that some videos can overlap, in particular videos 2 and 7. This can be explained by their relatively close streaming pattern as shown in Figure 3.2.

3.4 Summary

In this chapter, we present three deep learning models to classify encrypted WiFi video traffic. we investigated the possibility of discovering video-streaming content from passively observed WiFi traffic that is encrypted at both transport layer and MAC layer.

In order to unveil this WiFi video traffic, we proposed three types of neural networks, namely a Recurrent Neural Network and a Multi-Layer Perceptron, and a Convolutional Neural Network to analyze the captured traffic. Overall, we have demonstrated that by leveraging the particular DASH pattern of each video, the MLP model was able to achieve 97% accuracy in identifying videos from a closed set of 10 videos in encrypted WiFi traffic. This high accuracy was later re-evaluated when, two weeks after the original data collection, we collected a new set of data, and using the same original model we were able to obtain similar performance. Thus demonstrating the robustness of our approach.

Chapter 4

Classifying Encrypted WiFi Traffic Using A Hierarchical Classifier

4.1 Introduction

We introduced the models for video traffic classification in Chapter 3. Not only videos but also audios and websites need to be classified.

This work investigates the possibility of classifying traffic flows using a hierarchical architecture. It can make coarse-grained predictions (e.g. web, video, or audio) as well as fine granular predictions (e.g. content providers/platforms and exact content). Furthermore, we show that a hierarchical approach also allows the estimation of network usage characteristics for the purposes of service management in large networks, which constitutes a first step toward identifying unknown service providers for different traffic classes.

To summarize this chapter's work, we list the main contributions as follows:

- We extend the idea of content fingerprinting to build a hierarchical traffic classifier that is able to make coarse-grained and fine-grained predictions about encrypted traffic flows by leveraging weight sharing features in convolutional neural networks.
- We show that our hierarchical classifier can achieve over 95% accuracy in identifying traffic types such as web, video streaming, and audio streaming as well as identifying content providers of traffic and the exact content consumed by the user.

- we demonstrate its potential for classifying previously unseen content to its corresponding traffic type and content provider.

4.2 Method

As stated in Chapter 1, there are different traffic flows using HTTPS communication such as video, audio, and web traffic. It is challenging to classify these traffic flows especially when they are encrypted. In this section, we explain the design of a hierarchical traffic classifier that is able to identify different HTTPS content types at the higher level and at lower levels and identify different content providers/platforms and their exact content.

4.2.1 Streaming and Other Time Sensitive Traffic

As stated in Chapter 3, video streaming over the Internet has shifted to what is commonly referred to as HAS. Similarly, other types of encrypted web traffic have unique temporal patterns. We illustrate this in Figure 3.1 using example I/O graphs of each HTTPS content type from three platforms. We selected YouTube, Netflix and Stan as the three video platforms, YouTube Music, Spotify and Xiami as the audio platforms, Wikipedia, ABC and SMH as the Web websites, and we randomly choose some videos, songs and web pages.

As shown in Figure 3.1, video traffic fluctuates significantly due to DASH. Audio streaming starts with a large burst and steadies afterward. For web browsing, the traffic peaks can have larger time gaps corresponding to surfing patterns. Also, it is noticeable that for the same HTTPS content type, the traffic patterns are completely different across platforms due to the changes in encoding and the specifics of streaming protocols. For example, among video platforms, Netflix loads most of the data at the beginning of each playback, while YouTube loads data periodically. Similarly, Spotify loads data periodically during the playback, while Xiami loads everything at the start.

4.2.2 Architecture

We show the network architecture of our hierarchical classifier in Figure 4.1. The model consists of four parts: i) common module, ii) traffic type classification module, iii) content provider classification module, and iv) content classification module.

The common module consists of two convolution layers that extract the shallow features from the original data before being fed into next the classification modules. This allows the reduction of the number of parameters by weight sharing and has been used successfully in image classification tasks [119].

The traffic type classification module that classifies HTTPS traffic types (e.g. video, audio, and web), consists of 1 convolution layer, 1 max pooling layer, 1 fully connected layer, 1 dropout layer and 1 fully connected layer. The content provider classification module is used to classify the traffic source, i.e. the content platform where the traffic comes from. Based on the prediction of the traffic type classification module, the output of the common module is fed into the content provider classification module with corresponding weights which have been trained for video, audio and web text types separately. The architecture of this module is the same as the traffic type classification module. Similarly, the content classification module is utilized to classify the sub-classes based on the result from the content provider classification module. The output of the first CNN layer of the content provider classification module is fed into this module. This module is comprised of 2 convolution layers, 2 max pooling layers, 2 fully connected layers and 1 dropout layer. The filter size of convolution layers in our hierarchical classifier is 1×16 and each convolution layer has 32 filters. The filter size of max pooling layers is 1×6.

This architecture also allows handling previously unknown content towards an open-set classification. From our hierarchical approach, the classifier is able to identify at least the type of content if a new content provider is detected. For example, if a new video content platform emerges and thus has not been seen during

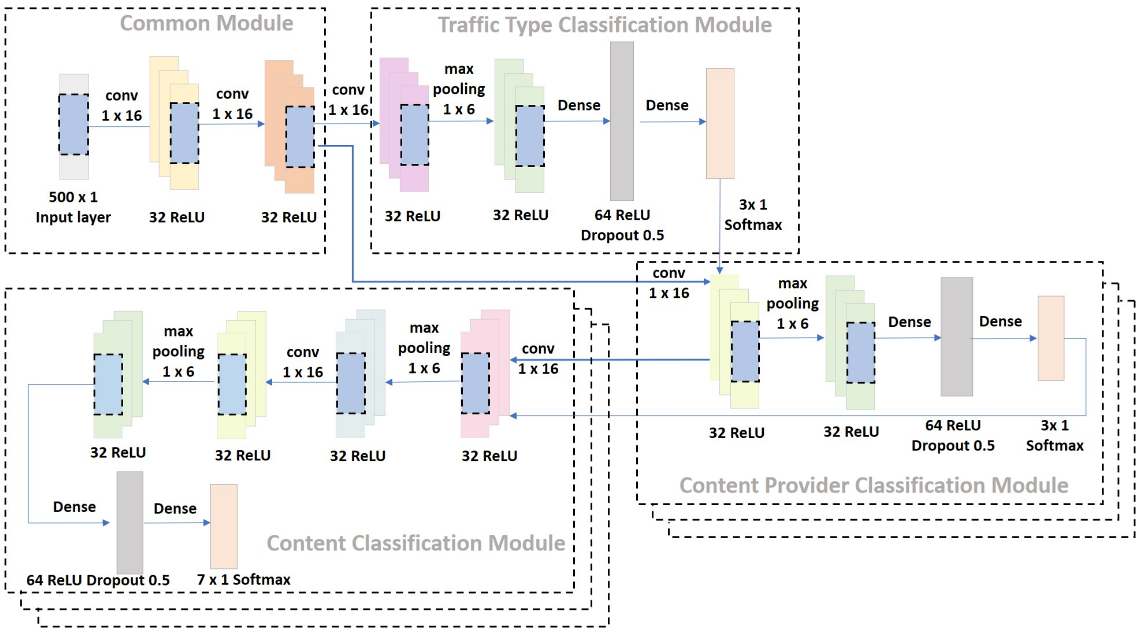


Figure 4.1 : Hierarchical classifier architecture

the training phase, our traffic type classification module will still classify the traffic as video because of the common characteristics of video traffic.

Here, we utilize a CNN module as the unit of the hierarchical classifier for three reasons. CNN models are known for their hierarchical feature extraction capabilities. For example, multiple works that studied the filter responses of image classifying CNNs [28; 33; 103] demonstrated that filters of the initial layers learn to identify high level patterns such as lines, and the filters in deep layers learn to identify more complex shapes such as people and texture. Such behavior would fit well into the structure of our dataset where different levels of granularities are required first to identify traffic type, next the platform, and finally the exact content. Besides, a CNN module can extract features through filters layer by layer from original data at different granularity levels which is suitable for our multiple level hierarchical structure. In our hierarchical classifier data set, there are video, audio, and web traffic, three types of traffic from different content providers. Thus, it is intuitive to utilize a CNN module to extract features on different granularity levels to identify the traffic type and content providers.

4.2.3 Training Process

As previously mentioned, our classifier model is divided into three levels. Corresponding to the three-level network structure, each sample has three labels, that is, traffic type label, content provider label, and content label.

We trained the whole model in three stages. In the first stage, training data and their traffic type labels are fed into the *common module* and the *traffic type classification module* to classify HTTPS traffic. After this training, the weights of the two modules are fixed. In the second stage, we separately fed video, audio and web page data flows through the pre-trained common module and then fed the output of the common module into the corresponding *content provider classification module* based on the classification result of the traffic type module. In this stage, we trained the content provider classification module with the corresponding label. At the final stage, we trained the *content classification module* by freezing the weights of the pre-trained module. The details are shown in Algorithm 1.

4.3 Experiments and Results

4.3.1 Dataset

We configured a laptop to connect to an 802.11n WiFi access point using channel 6 of the 2.4 GHz spectrum with WPA2 encryption. From this laptop, we repeatedly downloaded the same video, audio, and web browsing content. In particular, we first selected 30 different videos on Netflix, Stan, and YouTube; 10 songs on YouTube Music, Spotify, and Xiami; and 10 web page lists from Wikipedia, ABC News, and Sydney Morning Herald every three minutes.[*] In a university environment, we ac-

[*]The lists of videos, songs, and web pages are presented on https://cloudstor.aarnet.edu.au/plus/s/enerU4Gg54SShdC

Algorithm 1 Hierarchical training process

```
Required: $\mathbf{NN_{comm}}, \mathbf{NN_{type}}, \mathbf{NN_{provider}}, \mathbf{NN_{content}}$
Required: $K_{type}, K_{provider}$ = number of traffic type categories, number of content provider categories
Required: $\mathbf{X}, \mathbf{Y_1}, \mathbf{Y_2}, \mathbf{Y_3}$ =data, traffic type label, content provider label, content label
Update $\mathbf{NN_{comm}}$ and $\mathbf{NN_{type}}$ through minimizing cross-entropy loss:
    $L_{type} = -\hat{y_1} \log(NN_{type}(x))$
for k = 1 to $K_{type}$ do
    Update $\mathbf{NN_{provider\ k}}$ through minimizing cross-entropy loss:
        $L_{provider} = -\hat{y_2} \log(NN_{provider}(x))$
end for
for k = 1 to $K_{provider}$ do
    Update $\mathbf{NN_{content\ k}}$ through minimizing cross-entropy loss:
        $L_{content} = -\hat{y_3} \log(NN_{content}(x))$
end for
```

cessed the selected content via a web browser over 100 times and we captured the first three minutes of each stream in the eavesdropper computer. On this eavesdropper laptop, we used AirPcap Nx from Riverbed to passively capture all the frames available on this channel regardless of the Ethernet address within the frame. We note that in the case of YouTube Music and Video, we also used a premium account to remove advertisements. Furthermore, to simulate simple web surfing, we opened 18 URLs one by one at 10 seconds intervals to obtain one three-minutes sample of web surfing. Finally, we filtered the captured traffic to record only data frames from and to a target MAC address, which we can consider as a unique identifier and therefore that a single user corresponds to a given MAC address.

Overall, we collected two datasets. Dataset 1 contains 9,000 samples in total consisting of 3,000 samples each for video, audio, and web. And within each content provider of video, audio, and web, there are diverse sub-classes; that is, 10 videos, 10 songs, and 10 lists of web pages, respectively. Dataset 2 is an imbalanced dataset that contains 15,000 samples, in which 9,000 samples from video, 3,000 samples from audio and 3,000 samples from web pages. And within each content provider of video, audio, and web, there are 30 videos, 10 songs and 10 lists of web pages respectively. For dataset 1, We divided it following a 70-30 partition where 70% of data was used as the closed set. In other words, samples of 7 videos, 7 songs and 7 lists of web pages were used to train and test the hierarchical model in a closed set scenario and the other 30% of data which includes samples of 3 videos, 3 songs and 3 lists of web pages were used as unknown data to test the performance of our model akin to a more generic setup (also referred hereafter as real-world setup). For the *closed set classification*, we shuffle the 70% of captured traffic and then split it into training, development and testing sets in 60% :20% :20% ratio.

4.3.2 Evaluation Metrics

We adopt four popular classification metrics Accuracy, Precision, Recall, and F1-score for evaluation.

Accuracy represents the proportion of correctly predicted samples in all samples. It is the most intuitive evaluation metric in classification problems. But, when the proportion of samples of different categories is very unbalanced, the category with a large proportion tends to be the most important factor affecting the accuracy. In this case, we have to consider other evaluation metrics including precision, recall, and F1-score. Precision refers to the ratio of correctly classified positive samples to the number of samples predicted as positive by the classifier. This is a statistic focusing on the samples determined by the classifier to be positive. Recall refers to the ratio of correctly classified positive samples to the number of true positive samples. This is a statistic focusing on the real positive samples. Precision and recall are a trade-off relationship. F1 score is the harmonic mean of precision and recall.

Table 4.1 : Hierarchical model results on dataset 1 (in percent)

Feature	**F3-Downlink**				**F1-Combination**				**F6-Combination**			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Traffic Type	98.22	98.23	98.22	98.21	99.03	99.03	99.03	99.03	98.70	98.70	98.70	98.70
Content Provider Classification Module												
Video	97.54	97.71	97.54	97.54	97.05	97.08	97.05	97.05	98.53	98.54	98.53	98.52
Audio	98.28	98.37	98.28	98.29	99.51	99.5	99.51	99.51	99.75	99.76	99.75	99.75
Web	98.10	98.10	98.10	98.10	98.81	98.81	98.81	98.81	99.05	99.06	99.05	99.05
Content Classification Module												
YouTube (V)	96.35	96.36	96.35	96.34	95.62	95.77	95.62	95.65	95.62	95.66	95.62	95.61
Netflix	67.18	72.71	67.18	67.74	53.44	60.70	53.44	50.70	54.20	60.34	54.20	52.67
Stan	55.40	61.68	55.40	52.64	55.40	56.65	55.40	55.19	54.68	53.95	54.68	54.17
Spotify	24.03	19.71	24.03	20.99	17.83	20.11	17.83	8.72	24.03	22.28	24.03	19.95
Xiami	18.98	31.86	18.98	19.26	16.79	16.17	16.79	13.20	26.28	27.48	26.28	23.06
YouTube (M)	97.16	97.66	97.16	97.21	99.29	99.32	99.29	99.29	100	100	100	100
ABC	99.30	99.33	99.30	99.30	98.59	98.62	98.59	98.59	99.29	99.33	99.30	99.30
WikiPedia	88.89	89.10	88.89	88.78	96.53	97.04	96.53	96.58	95.83	96.52	95.83	95.82
SMH	97.76	97.80	97.76	97.74	94.03	94.76	94.03	93.98	97.02	97.10	97.02	97.01

Table 4.2 : Hierarchical model results on dataset 2 (in percent)

Feature	**F3-Downlink**				**F1-Combination**				**F6-Combination**			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
Traffic Type	99.46	99.46	99.46	99.46	99.42	99.42	99.42	99.42	98.88	98.88	98.88	98.88
Content Provider Classification Module												
Video	98.05	98.09	98.05	98.06	98.11	98.13	98.11	98.11	97.30	97.30	97.30	97.29
Audio	99.02	99.02	99.02	99.02	99.51	99.51	99.51	99.51	99.75	99.76	99.75	99.75
web	98.33	98.34	98.33	98.33	99.29	99.29	99.29	99.29	99.05	99.06	99.05	99.05
Content Classification Module												
YouTube (V)	80.45	81.51	80.45	80.19	64.43	69.30	64.43	61.87	72.63	74.47	72.63	72.76
Netflix	87.57	88.37	87.57	87.29	71.65	77.19	71.65	70.89	62.14	64.08	62.15	61.78
Stan	49.54	54.02	49.54	49.27	46.75	52.51	46.75	46.47	33.40	32.98	32.98	32.98
Spotify	27.13	23.14	23.14	24.052	20.16	8.12	20.16	10.44	20.38	21.19	19.380	12.70
Xiami	24.09	47.17	24.09	16.25	18.25	36.10	18.25	14.72	16.06	17.52	16.06	11.58
YouTube (M)	85.11	87.46	85.11	85.14	100	100	100	100	98.58	98.58	98.58	98.57
ABC	99.30	99.33	99.30	99.30	100	100	100	100	98.59	98.66	98.59	98.59
WikiPedia	95.14	95.49	95.14	95.12	98.61	98.68	98.61	98.61	97.22	97.27	97.22	97.22
SMH	98.51	98.58	98.51	98.50	99.25	99.30	99.25	99.25	97.76	97.93	97.76	97.79

4.3.3 Implementation Details

We implement our model with Tensorflow. The batch size is set as 64. We use Adam optimizer and set the learning rate as 0.0001. The number of training epochs is 300.

4.3.4 Results

We present the performances of our hierarchical classifier on dataset 1 and dataset 2, respectively. On dataset 1, we observed that for the first level, data flows are identified as video, audio or web page with an accuracy of 99.0%. For the second level, we can further identify from which content provider the flows come. For video flows, it can be identified as YouTube, Netflix or Stan with accuracy of 97.5% as well as audio and web page flow can be identified as Spotify, Xiami, Youtube Music and ABC, SMH, Wikipedia with an accuracy of 98.3% and 98.1% respectively. Finally, at the content level, our model identified the deep content of flows with an accuracy for YouTube, Netflix, Stan, Spotify, Xiami, YouTube music, ABC, WikiPedia and SMH of 96.4%, 67.2%, 55.4%, 24.0%, 19.0%, 97.2%, 99.3%, 88.9% and 97.8% respectively.

In dataset 2, we observed that at the first level, the accuracy can reach 99.4%. At the second level, the accuracy of video, audio and web can reach 98.1%, 99.0% and 98.3% respectively. Finally, at the content level, our model identified the content of flows with accuracy for YouTube, Netflix, Stan, Spotify, Xiami, YouTube Music, ABC, Wikipedia and SMH of 80.5%, 87.6%, 49.5%, 27.1%, 24.1%, 85.1%, 99.3%, 95.1% and 98.5% respectively.

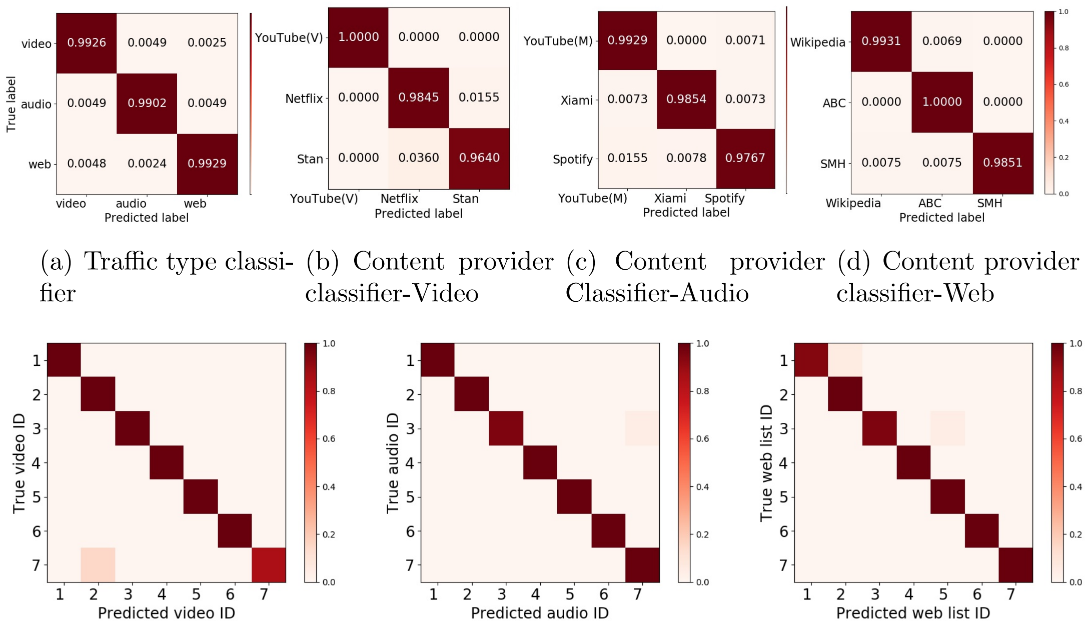
In practice, it is difficult to create a classifier for all existing traffic classes, as well as keep up to date with any new content providers. This problem, known as the open-set problem, cannot be tackled directly with our hierarchical approach. However, our approach could mitigate the problem, at traffic type and content provider levels. For example, our approach can identify video content from unknown providers. Similarly, we can map unknown content to the correct content provider platform. To demonstrate this possibility, we tested our model with 3 videos, 3 songs and 3 web page lists that were never shown at training time. Our hierarchical model could identify content provider with a very high accuracy (average of 96.3%). The individual content provider identification accuracies were: YouTube - 95.3%, Netflix - 98.5%, Stan - 100%, Spotify - 90.1%, Xiami - 100%, YouTube Music - 99.3 Wikipedia - 96.4%, ABC - 74.3%, and SMH - 98.0%.

4.3.5 Result Analysis

These results are elaborated in the form of confusion matrices. From this figure, we can see the classification capability of traffic type classifier is balanced on video, audio and web traffic types with an accuracy of over 99%. For content provider classifiers, the classification performs unevenly depending on the content provider. The classification performance is detailed in Table 4.1, where content classification accuracy on Stan and Netflix is lower than on YouTube. From the confusion matrix of content classifiers, we also found that some of the videos (7th video on YouTube) are consistent with the ones, as well as audios (3rd audio on YouTube music) and web pages (1st and 3rd web pages on wiki) are misclassified to other classes with higher probability.

Next, we analyze the performance of content classification, which varies significantly compared to content provider classification.

For video, the flows from YouTube perform significantly better than those from Netflix and Stan. The main reason for this result is that video data from YouTube is collected primarily from the first three minutes of each YouTube video run. However, Netflix and Stan data are not entirely extracted from the first three minutes of each video run because Netflix and Stan platforms (websites) play videos from the



(a) Traffic type classifier (b) Content provider classifier-Video (c) Content provider Classifier-Audio (d) Content provider classifier-Web

(e) Content classifier-YouTube (f) Content classifier-YouTube Music (g) Content classifier-Wiki

Figure 4.2 : Classification performance confusion matrix for hierarchical models with F6-combination

previous playing segment.

As shown in Figure 4.3, the pattern of the same Stan video on different runs is different. That explains why the classification performance of Netflix and Stan videos is unsatisfactory. We plan to investigate in future work the full extent of the consequences induced by this data collection by comparing the results with new capture data.

For audio, the YouTube Music classifier performs reasonably well. However, classifiers for Spotify and Xiami do not achieve favorable results. As shown in Figure 4.4, there is a characteristic of Spotify and Xiami traffic flows that they only have one or few peaks at the beginning, which means the information that can be used to identify the specific song is limited. But for YouTube music, the traffic flows not only have some peaks in the beginning but also have some fluctuation in the rest of the flows, which explains why only the audio flows from YouTube Music can be

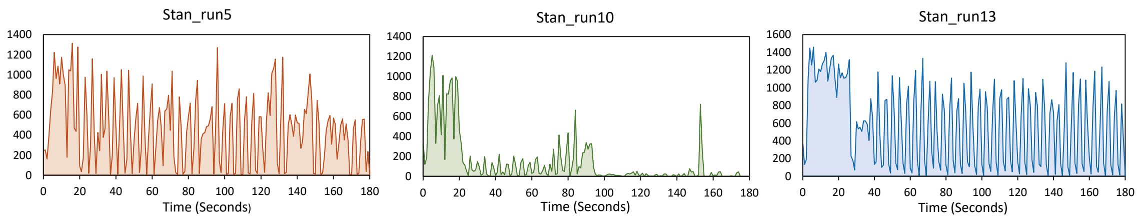


Figure 4.3 : I/O graphs of the same Stan video on different runs

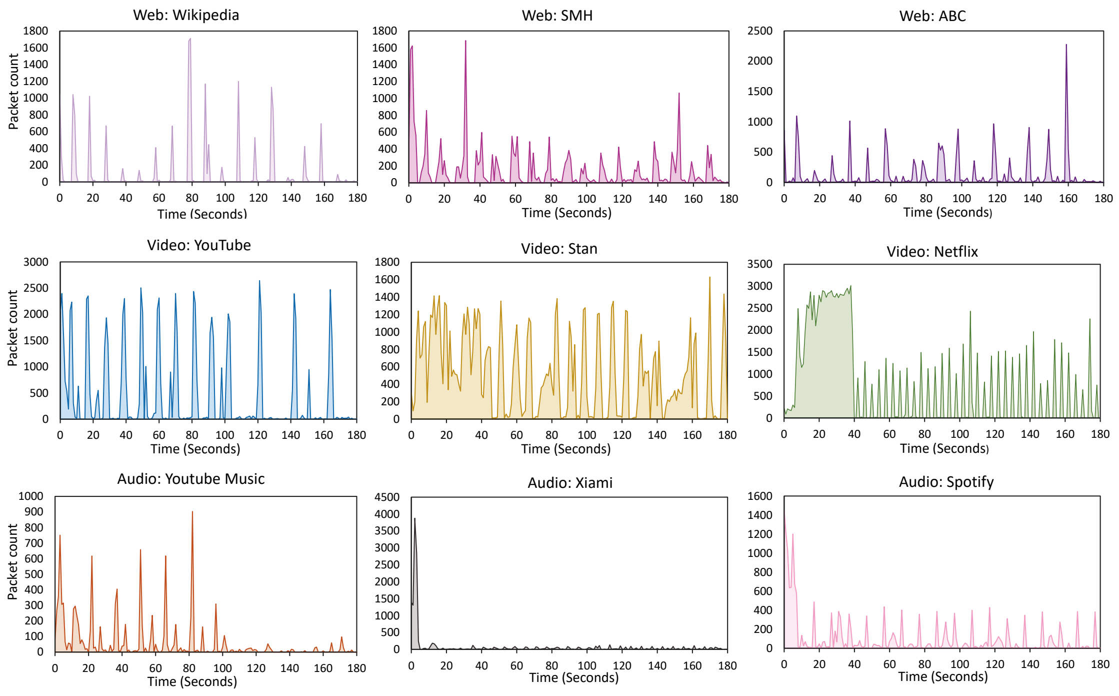


Figure 4.4 : I/O graphs of different HTTP traffic types on different content providers.

classified with high accuracy. In terms of content size, audio is significantly smaller compared to video, as a result, service providers tend to download the requested song rather than stream similar to a video. Xiami and Spotify I/O graph in Figure 4.4 depicts this behavior. As a result, the proposed hierarchical classifier failed to identify Xiami and Spotify audio content. In contrast, YouTube music still achieves a very high accuracy as they are streamed in real-time similar to YouTube videos.

For web pages, the performance of Wikipedia, ABC and SMH are outstanding with an accuracy of around 99%. However, this accuracy might be induced by our browsing method. Indeed, we use a simple way to simulate web browsing, that is, opening 18 web pages at 10 seconds interval every 3 minutes. As shown in Figure 4.4, the content of web page generates distinct traffic signatures. Therefore, it is possible to identify the list of web pages according to the flow pattern. However, in a real situation, the dynamic web content such as advertisement frames of web pages will result in an uncertain flow pattern and, consequently, increase the difficulty of identification.

For the experiment on dataset 2, we noticed that, at the first and second levels, the performance remains at the same level as experiment 1. Nonetheless, we found that, for video and audio content classification (i.e., the third level), the accuracy decreased by approximately 10%-15%. The flows from YouTube perform worse than experiment 1. The main reason for this result is that the content of the additional 20 YouTube videos is similar (i.e, they all belong to the nature category). Nonetheless, in practice, this can be resolved by changing the model architecture at the third level (e.g., by adding more layers) so that the model can handle a large number of classes.

Finally, we highlight that, dataset 2 is unbalanced, in which the video data is three times the number of the audio or web data. However, we noticed that at the first and second levels, the values of precision, recall and F1 score were very close to each other indicating that our model has a better balance between precision and recall.

To summarize, we can conclude that our hierarchical model has excellent performance for both traffic type classifier and content provider classifier. For content classifiers, video and web page flows can be correctly classified with good performance, but it does not classify well for audio flows except for those flows from YouTube Music.

4.4 Summary

In this chapter, we showed that it is possible to build a hierarchical traffic classifier that can identify traffic type, content provider, and exact content by passively observing encrypted traffic flows. We showed the feasibility of this approach for the most common traffic types on the internet: web, video, and audio. Our approach is suitable for targeted surveillance applications. For example, law enforcement can use a solution similar to ours to monitor user activities in an apartment block by passively observing encrypted WiFi traffic. They will be able to isolate suspects who may be visiting a specific website or watching a specific video. A similar hierarchical approach can also be used to make inferences about target user activities when they are using VPNs. VPNs add an extra layer of encryption (analogous to what is happening in WiFi) and few works already demonstrated single-level traffic classification over VPN traffic [52; 114]. Such work can be extended using the hierarchical approach we propose here.

Chapter 5

Generating Samples by Category Using Bayesian Nonparametric Autoencoders

This chapter aims to achieve our *objective*③: generating samples by category using Bayesian nonparametric based autoencoders.

5.1 Introduction

As introduced in Chapter 1, the quality and quantity highly affect the performance of deep learning classifiers. Generative models are wildly used for data argumentation. The generated samples can add to the training process to increase the performance of the classifier. In this chapter, we introduce two novel generative models, namely IGMVAE and I^{2}GMVAE.

VAE is an important generative model, in which the prior distribution of latent variables is a standard Gaussian distribution. However, using the same prior distribution of different categories makes it hard to generate data for a certain category. In fact, latent variables sometimes contain category information. For example, if we use MNIST data [81] to train the VAE, we can observe latent variables of different digits actually from different clusters in the latent variable space. GMVAE [55] replaces the original distribution with a Gaussian mixture model(GMM) [104]. For GMVAE, the number of classes needs to be given before generating the data [32]. But, in some cases, we do not have labeled data (including the number of categories). To the best of our knowledge, there are currently no generative models for this situation. As mentioned in chapter 2, BNP approaches estimate the number of categories from the observed data. So we do not need to specify the number of categories in advance. Besides, BNP models allow future data to exhibit previously unseen clusters. So that when new data is added, the model can automatically do an adjustment to adapt to the new data.

IGMM [89] and I^{2}GMM [120] are two type of classic BNP models. Based on that, we propose two variants of the VAE with IGMM and I^{2}GMM as the prior distribution of latent space.

To verify the effectiveness of our models, we do experiments on a YouTube traffic dataset [65]. The experiment results show that compared with GMVAE, our model achieves a better clustering and generation effect. And, our models generate data by category without labeled information in the training process, while the number of categories should be specified in the GMVAE model. Moreover, when there are new data added in, our models have the ability to adjust the cluster number and generate the new clusters' samples.

To summarize this chapter's work, we list the main contributions as follows:

- We propose two novel generative models based on VAE and BNP models.
- We test our models in a YouTube video dataset. The results show that our models can generate samples by category without label information added in the training process.
- We compare our model with a state-of-art GMVAE proposed for the same generation task [27]. We show that our models can achieve a better clustering and generation effect.
- We explored the possibility that our models can adapt to incremental data and scale to new categories.

5.2 IGMVAE

In the thesis, we propose a novel generated model, namely IGMVAE. In this section, we explain the design of our IGMVAE model.

5.2.1 Method

Based on VAE, we replace the Gaussian distribution with IGMM as the prior distribution over latent variables. we choose the IGMM as priors because it is an extension of Gaussian distribution and GMM. We assume the data point is from an IGMM, then infer the data point is from which underlying distribution pattern. IGMM learns the cluster number by itself. So that we do not have to specify the number of categories like GMM. Besides, IGMM will assign samples that do not belong to the training set to new clusters. This characteristic allows our models to do incremental learning.

Consider the generative model

$$p(y, x, z, \pi) = \prod_{i}^{N} p(y_i|x_i)p(x_i|z_i)p(z_i|\pi) \prod_{k}^{\infty} p(\pi_k|\alpha) \tag{5.1}$$

where an observed sample y is generated under the following process:

$$\begin{aligned} v_k &\sim \text{Beta}(1, \alpha) \\ \pi_k &= v_k \prod_{j=1}^{k-1}(1 - v_j) \\ G &= \sum_{k=1}^{\infty} \pi_k \delta_{w_k} \\ \text{for each sample} \quad i &: \\ z_i &\sim \mathcal{C}(\{\pi\}) \\ x_i &\sim \mathcal{N}(\mu_x(z_i, \theta), \sigma_x(z_i, \theta)) \\ y_i &\sim \mathcal{N}(\mu_y(x_i, \phi), \sigma_y(x_i, \phi)) \end{aligned} \tag{5.2}$$

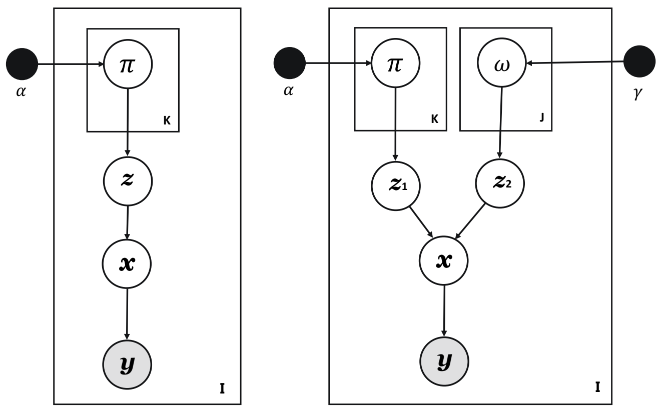


Figure 5.1 : Probabilistic graphical models for the IGMVAE(left) and I^{2}GMVAE(right).

where y is generated from a neural network with parameter ϕ and the input x. And μ_y, and σ_y are calculated by this neural network. Furthermore, $p(x|z)$ is an IGMM specified by another neural network model parameterized by θ and with input z. μ_y,and σ_y are calculated by this neural network. z is a one-hot vector sampled from π, which means which component has been chosen from IGMM. The generation probability graph is showed in Figure 5.1.

5.2.2 Inference Process

We truncated at K, it means that $v_K = 1$. The variational probability distributions are defined as

$$\begin{aligned}
q(v_k) &= \text{Beta}(\tau_{v,k,1}, \tau_{v,k,2}), \text{for} \quad k = 1 : K-1 \\
q(\pi_k) &= \delta(v_k \prod_{i=1}^{k-1}(1 - v_i)) \\
&\text{for each sample} \quad i : \\
q(z_i|y_i) &= \mathcal{C}(\mu_z(y_i, \nu), \sigma_z(y_i, \nu)) \\
q(x_i|y_i) &= \mathcal{N}(\mu_x(y_i, \delta), \sigma_x(y_i, \delta))
\end{aligned} \tag{5.3}$$

Where x is estimated by a neural network model parameterized by δ and y. z is estimated by a neural network model parameterized by ν and y. The last layer of this neural network is Gumbel softmax [50].

The model is trained by optimizing the ELOB, which can be written as:

$$\mathcal{L}_{elbo} = \mathbf{E}\left[\frac{p(y,x,z,\pi)}{q(x,z,\pi|y)}\right] \tag{5.4}$$

Applying independence assumption, we get the decomposition:

$$q(x,z,\pi|y) = \prod_{i}^{N} q(x_i|y_i)q(z_i|y_i)\prod_{k}^{K} q(\pi_k) \tag{5.5}$$

The low bound can be then written as:

$$\begin{aligned}\mathcal{L}_{elbo} =& \mathbf{E}_q\left[\log p(y|x)\right] - \mathbf{KL}\left[q(x|y)||p(x|z)\right] \\ & - \mathbf{KL}\left[q(z|y)||p(z|\pi)\right] - \mathbf{KL}\left[q(\pi)||p(\pi)\right]\end{aligned} \tag{5.6}$$

The terms in the ELOB includes reconstruction loss term, x KL-divergence loss term, z KL-divergence loss term and π KL-divergence loss term respectively.

Where the reconstruction loss term can be computed by the corresponding loss function,

$$\mathbf{E}_q\left[\log p(y|x)\right] = -ylog(\hat{y}) - (1-y)log(1-\hat{y}) \tag{5.7}$$

The x KL-divergence loss term can be calculated by the KL distance formula of two Gaussian distributions,

$$\mathbf{KL}\left[q(x|y)||p(x|z)\right] = -\frac{1}{2}[\log\frac{\sigma_o(y)}{\sigma_x(z)} - \frac{\sigma_o(y)}{\sigma_x(z)} - \frac{\mu_o(y)-\mu_x(z)}{\sigma_x(z)} + 1] \tag{5.8}$$

The z KL-divergence loss term can be calculated by the KL distance formula,

$$
\begin{aligned}
&\mathbf{KL}\left[q(z|y)||p(z|\pi)\right]\\
=&\mathbf{E}_q[\log\frac{q(z)}{p(z)}]\\
=&\mathbf{E}_q\left[\sum_{k=1}^{K} q(z|y)_k\left[\log q(z|y)_k-\log\pi_k\right]\right]\\
=&\sum_{k=1}^{K} q(z|y)_k\log q(z|y)_k-\mathbf{E}_q\left[\sum_{k=1}^{K} q(z|y)_k\log\pi_k\right]\\
=&\sum_{k=1}^{K} q(z|y)_k\log q(z|y)_k-\sum_{k=1}^{K} q(z|y)_k\mathbf{E}_{q(\pi)}\log\pi_k\\
=&\sum_{k=1}^{K} q(z|y)_k\log q(z|y)_k\\
&-\sum_{k=1}^{K} q(z|y)_k\left[\mathbf{E}_{q(\pi)}\log v_k+\sum_{j=1}^{k-1}\mathbf{E}_{q(\pi)}\log(1-v_j)\right]\\
=&\sum_{k=1}^{K} q(z|y)_k\left[\log q(z|y)_k\right]\\
&-\sum_{k=1}^{K} q(z|y)_k\left[\psi(\tau_{k,1})-\psi(\tau_{k,1}+\tau_{k,2})+\sum_{j=1}^{k-1}(\psi(\tau_{j,2})-\psi(\tau_{j,1}+\tau_{j,2}))\right]
\end{aligned}
\tag{5.9}
$$

The π KL-divergence loss term is calculated in the following,

$$
\mathbf{KL}\left[q(\pi)||p(\pi)\right]=\left[\sum_{k=1}^{K}\mathbf{E}_q\log q(\pi_k)-\sum_{k=1}^{K}\mathbf{E}_q\log p(\pi_k)\right]
\tag{5.10}
$$

where,

$$\begin{aligned}
&\mathbf{E}_q \log p(\pi_k) \\
=&\mathbf{E}_q \left[\log p(v_k) + \sum_{j=1}^{k-1} \log p(1 - v_j)\right] \\
=&\mathbf{E}_q \log p(v_k) + \sum_{j=1}^{k-1} \mathbf{E}_q \left[\log p(1 - v_j)\right] \\
=&\mathbf{E}_q \left[\log \frac{(1 - v_k)^{\alpha-1}}{\mathbf{B}(1, \alpha)}\right] + \sum_{j}^{k-1} \mathbf{E}_q \left[\log \frac{(1 - v_j)^{\alpha-1}}{\mathbf{B}(\alpha, 1)}\right] \\
=&\mathbf{E}_q \left[(\alpha - 1) \log(1 - v_k) - \log \mathbf{B}(1, \alpha)\right] \\
&+ \sum_{j=1}^{k-1} \mathbf{E}_q \left[(\alpha - 1) \log(1 - v_j) - \log \mathbf{B}(\alpha, 1)\right] \\
=&(\alpha - 1)(\psi(\tau_{k,2}) - \psi(\tau_{k,2} + \tau_{k,1})) - \log \mathbf{B}(1, \alpha) \\
&+ \sum_{j=1}^{k-1} \left[(\alpha - 1)(\psi(\tau_{j,2}) - \psi(\tau_{j,2} + \tau_{j,1})) - \log \mathbf{B}(\alpha, 1)\right]
\end{aligned} \tag{5.11}$$

where,

$$\begin{aligned}
&\mathbf{E}_q \log q(\pi_k) \\
=&\mathbf{E}_q \left[\log q(v_k) + \sum_{j=1}^{k-1} \log q(1 - v_j)\right] \\
=&\mathbf{E}_q \log q(v_k) + \sum_{j=1}^{k-1} \mathbf{E}_q \left[\log q(1 - v_j)\right] \\
=&\mathbf{E}_q \log \left[\frac{v_k^{\tau_{k,1}-1}(1 - v_k)^{\tau_{k,2}-1}}{\mathbf{B}(\tau_{k,1}, \tau_{k,2})}\right] + \sum_{j}^{k-1} \mathbf{E}_q \left[\log \frac{(1 - v_j)^{\tau_{j,2}-1} v_j^{\tau_{j,1}-1}}{\mathbf{B}(\tau_{j,2}, \tau_{j,1})}\right] \\
=&\mathbf{E}_q \left[(\tau_{k,1} - 1) \log v_k + (\tau_{k,2} - 1) \log(1 - v_k) - \log \mathbf{B}(\tau_{k,1}, \tau_{k,2})\right] \\
&+ \sum_{j=1}^{k-1} \mathbf{E}_q \left[(\tau_{j,1} - 1) \log v_j + (\tau_{j,2} - 1) \log(1 - v_j) - \log \mathbf{B}(\tau_{j,2}, \tau_{j,1})\right] \\
=&(\tau_{k,1} - 1)\mathbf{E}_q \log v_k + (\tau_{k,2} - 1)\mathbf{E}_q \log(1 - v_k) - \log \mathbf{B}(\tau_{k,1}, \tau_{k,2}) \\
&+ \sum_{j=1}^{k-1} [(\tau_{j,1} - 1)\mathbf{E} \log v_j + (\tau_{j,2} - 1)\mathbf{E} \log(1 - v_j) - \log \mathbf{B}(\tau_{j,2}, \tau_{j,1}]
\end{aligned}$$

$$
\begin{aligned}
=&(\tau_{k,1}-1)\left[\psi(\tau_{k,1})-\psi(\tau_{k,1}+\tau_{k,2})\right]+(\tau_{k,2}-1)\left[\psi(\tau_{k,2})-\psi(\tau_{k,2}+\tau_{k,1})\right]\\
&-\log\mathbf{B}(\tau_{k,1},\tau_{k,2})\\
&+\sum_{j=1}^{k-1}\left[(\tau_{j,1}-1)[\psi(\tau_{j,1})-\psi(\tau_{j,1}+\tau_{j,2})]+(\tau_{j,2}-1)\left[\psi(\tau_{j,2})-\psi(\tau_{j,2}+\tau_{j,1})\right]\right.\\
&-\log\mathbf{B}(\tau_{j,2},\tau_{j,1}) \qquad (5.12)
\end{aligned}
$$

The above is the calculation process of the ELOB which optimizes our model in the following experiments.

5.2.3 Architecture

In this section, we introduce the architecture of our model. The whole model contains an encoder and a decoder.

The encoding process consists of two neural networks:

- The label information z generated model
  $q(z_i|y_i) = \mathcal{C}(\mu_z(y_i,\nu),\sigma_z(y_i,\nu))$.
- The hidden variable x generated model
  $q(x_i|y_i) = \mathcal{N}(\mu_x(y_i,\delta),\sigma_x(y_i,\delta))$.

where $\mathcal{C}(\cdot)$ is Gumbel-softmax function[50].

The decoding process consists of two neural networks:

- The sample y reconstructed model
  $p(y_i|x_i) = \mathcal{N}(\mu_y(x_i,\phi),\sigma_y(x_i,\phi))$.
- The hidden variable x reconstructed model
  $p(x_i|z_i) = \mathcal{N}(\mu_x(z_i,\theta),\sigma_x(z_i,\theta))$.

These neural networks architecture is shown in Figure 5.2.

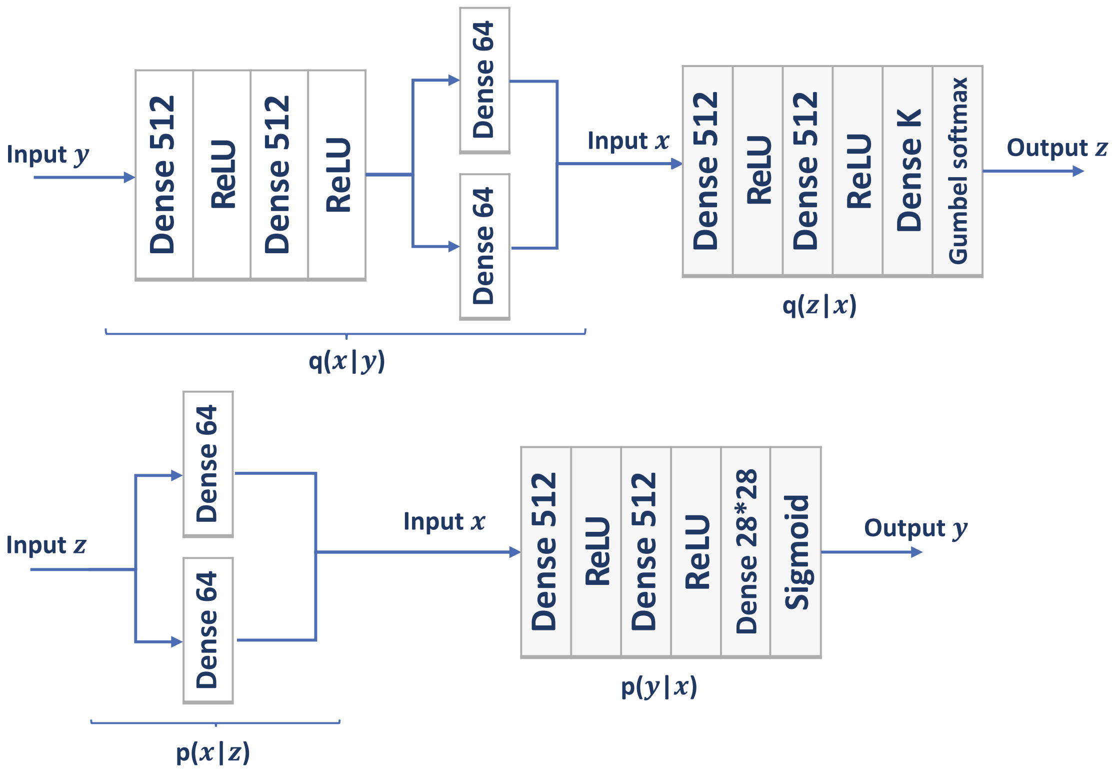


Figure 5.2 : The neural network architecture of IGMVAE.

5.2.4 Training and Testing

We use four loss terms in the training stage, which are $\mathcal{L}_{rec}$, $\mathcal{L}_{gaus}$, $\mathcal{L}_{categ}$ and $\mathcal{L}_{stickbreaking}$, corresponding to the reconstruction loss term, x KL-divergence loss term, z KL-divergence loss term and π KL-divergence loss term in ELOB respectively. Below we give the pseudo-code about how the updates are performed on the IGMVAE model.

As shown in Algorithm 2, the parameters are updated by optimizing the whole loss. At the test stage, the cluster number is learned by giving a threshold to π.

5.3 I^{2}GMVAE

As introduced above, we use IGMM as the prior distribution of latent variables. Sometimes, the data points that form different categories are very similar (e.g. dig-

Algorithm 2 Training process of IGMM

```
Required: Traffic dataset $Y$
Required: $\alpha$, $K$
Feed forward $y$ to $q(x|y)$ and $q(z|y)$,and get latent variable $x$ and $z$ .
Feed forward latent variable $x$ and $z$ to $p(y|x)$ and $p(x|z)$, and get faked sample $y'$.
Calculate the whole loss
    $$\mathcal{L}_{igmvae} = \mathcal{L}_{rec} + \mathcal{L}_{gaus} + \mathcal{L}_{categ} + \mathcal{L}_{stickbreaking}$$
Update parameter $\tau$, encoder parameters $q(x|y)$, $q(z|y)$ and decoder parameters $p(y|x)$, $p(x|z)$ with $\mathcal{L}_{igmvae}$.
```

ital 3 and digital 8). For this problem, we propose using I^{2}GMM as the prior distribution over latent variables. I^{2}GMM is an extension of IGMM, which is more suitable for more flexible modeling of datasets with skewed and multimodal cluster distributions. Unlike IGMM, which uses a single Gaussian per cluster, I^{2}GMM uses a single IGMM per cluster. It seems like a two-level IGMM, which has a better clustering effect on the dataset.

5.3.1 Method

Consider the generative model

$$p(y, x, z_1, z_2, \omega, \pi) = \prod_{i}^{N} p(y_i|x_i)p(x_i|z_{i,1}, z_{i,2})p(z_{i,1}|\pi)p(z_{i,2}|\omega, z_{i,1}) \prod_{k}^{K} p(\pi_k) \prod_{j}^{J} p(\omega_{k,j}) \tag{5.13}$$

where an observed sample y is generated under the following process:

$$
\begin{aligned}
v_k &\sim \text{Beta}(1, \alpha) \\
\pi_k &= v_k \prod_{j=1}^{k-1} (1 - v_j) \\
\langle \mu_k, \Sigma_k \rangle &\sim \mathcal{NIW}(0, I, \kappa_0, m_0) \\
G &= \sum_{k=1}^{\infty} \pi_k \delta_{\langle \mu_k, \Sigma_k \rangle} \\
\text{for each} \quad & k : \\
H_k &= \mathcal{N}(\mu_k, \Sigma_k / \kappa) \\
v_{k,j} &\sim \text{Beta}(1, \gamma) \\
\omega_{k,j} &= v_{k,j} \prod_{h=1}^{j-1} (1 - v_{k,h}) \\
w_{k,j} &\sim H_k \\
G_k &= \sum_{j=1}^{\infty} \omega_{k,j} \delta_{w_{k,j}} \\
\text{for each image} \quad & i : \\
z_{i,1} &\sim \text{Categorical}(\{\pi\}) \\
z_{i,2} &\sim \text{Categorical}(\{\omega_{z_{i,1}}\}) \\
x_i &\sim \mathcal{N}(\mu_x(z_{i,1}, z_{i,2}, \theta), \sigma_x(z_{i,1}, z_{i,2}, \theta)) \\
y_i &\sim \mathcal{N}(\mu_y(x_i, \phi), \sigma_y(x_i, \phi))
\end{aligned}
\tag{5.14}
$$

where y is generated from a neural network with parameter ϕ and the input x. And μ_y,and σ_y are calculated by this neural network. $p(x|z_1, z_2)$ is an I^{2}GMM specified by another neural network model parameterized by θ and with input z_1 and z_2 . z_1 is a one-hot vector sampled from π, which means which IGMM component has been chosen from I^{2}GMM. z_2 is a one-hot vector sampled from ω, which means which Gaussian component has been chosen from the corresponding IGMM component.

The generation probability graph is showed in Figure 5.1.

5.3.2 Inference Process

We truncated at K, it means that $v_K = 1$. And at J, it means that $v_{k,J} = 1$ The variational probability distributions are defined as,

$$\begin{aligned}
q(v_k) =& \text{Beta}(\tau_{k,1}, \tau_{k,2}), \text{for} \quad k = 1 : K - 1 \\
q(\pi_k) =& \delta(v_k \prod_{i=1}^{k-1}(1 - v_i)) \\
q(v_{k,j}) =& \text{Beta}(\zeta_{k,j,1}, \zeta_{k,j,2}), \text{for} \quad k = 1 : K, j = 1 : J - 1 \\
q(\omega_{k,j}) =& \delta(v_{k,j} \prod_{i=1}^{j-1}(1 - v_{k,i}))
\end{aligned} \tag{5.15}$$

for each sample $\quad i$:

$$\begin{aligned}
q(z_{i,1}|y_i) =& \mathcal{C}(\mu_{z,1}(y_i, \nu), \sigma_{z,1}(y_i, \nu)) \\
q(z_{i,2}|y_i) =& \mathcal{C}(\mu_{z,2}(y_i, \xi), \sigma_{z,2}(y_i, \xi)) \\
q(x_i|y_i) =& \mathcal{N}(\mu_x(y_i, \delta), \sigma_x(y_i, \delta))
\end{aligned}$$

Where x is estimated by a neural network model parameterized by δ and y. z_1 is estimated by a neural network model parameterized by ν and y. z_2 is estimated by a neural network model parameterized by ξ and y.

The generative model is trained with the variational inference objective, which can be written as:

$$\mathcal{L}_{elbo} = \mathbf{E}\left[\frac{p(y, x, z_1, z_2, \pi, \omega)}{q(x, z_1, z_2, \pi, \omega|y)}\right] \tag{5.16}$$

Applying the independence assumption, we get the decomposition:

$$\begin{aligned}
&q(x, z_1, z_2, \pi, \omega|y) \\
=& \prod_i^N p(y_i|x_i)p(x_i|z_{i,1}, z_{i,2})p(z_{i,1}|\pi)p(z_{i,2}|\omega, z_{i,1}) \quad \prod_k^K \left[p(\pi_k|\alpha) \prod_j^J p(\omega_{k,j}|\gamma)\right]
\end{aligned} \tag{5.17}$$

The low bound can be then written as:

$$\begin{aligned}\mathcal{L}_{elbo} &= \mathbf{E}_q\left[\log p(y|x)\right] - \mathbf{KL}\left[q(x|y)||p(x|z_1, z_2)\right] \\ &- \mathbf{KL}\left[q(z_1|y)||p(z_1|\pi)\right] - \mathbf{KL}\left[q(z_2|y)||p(z_2|\omega, z_2)\right] \\ &- \mathbf{KL}\left[q(\pi)||p(\pi)\right] - \mathbf{KL}\left[q(\omega)||p(\omega)\right]\end{aligned} \tag{5.18}$$

The terms in the ELOB include the reconstruction loss term, x KL-divergence loss term, z_1 KL-divergence loss term, z_2 KL-divergence loss term, π KL-divergence loss term, and ω KL-divergence loss term respectively.

The x KL-divergence loss term can be calculated by the KL distance formula of two Gaussian distributions, the derived process is the same as equation 5.8. The z_1 KL-divergence loss term's derived process is the same as equation 5.9. The π KL-divergence loss term's derived process is the same as equation 5.10. For the variable z_2, the derivation is as follows:

$$\begin{aligned}&\mathbf{KL}\left[q(z_2|y)||p(z_2|\omega, z_1)\right] \\ =&\mathbf{E}_{q(z_2,\omega,z_1)} \log \frac{q(z_2)}{p(z_2)} \\ =&\sum_{j=1}^{J} q(z_2|y)_j \log q(z_2|y)_j \\ &- \mathbf{E}_{q(\omega,z_2)}\left[\sum_{k=1}^{K} q(z_1|y)_k \log q(\omega_k|z_2)\right] \\ =&\sum_{j=1}^{J} q(z_2|y)_j \log q(z_2|y)_j \\ &- \sum_{k=1}^{K} q(z_1|y)_k \mathbf{E}_{q(\omega,z_2)} \log q(\omega_k|z_2) \\ =&\sum_{j=1}^{J} q(z_2|y)_j \log q(z_2|y)_j \\ &- \sum_{k=1}^{K} q(z_1|y)_k \mathbf{E}_{q(\omega)}\left[\sum_{j=1}^{J} q(z_2|y)_j \log \omega_{k,j}\right]\end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^{J} q(z_2|y)_j \log q(z_2|y)_j \\
&\quad - \sum_{k=1}^{K} q(z_1|y)_K \left[\sum_{j=1}^{J} q(z_2|y)_j \mathbf{E}_{q(\omega)} \log \omega_{k,j} \right]
\end{aligned} \tag{5.19}$$

where,

$$\mathbf{E}_{q(\omega)} \log \omega_{k,j} = \psi(\zeta_{k,j,1} + \zeta_{k,j,2})] + \sum_{l=1}^{j-1} [\psi(\zeta_{k,l,2}) - \psi(\zeta_{k,l,1} + \zeta_{k,l,2})] \tag{5.20}$$

For the variable ω, the derivation is as follows:

$$\begin{aligned}
&\mathbf{KL}\left[q(\omega)||p(\omega)\right] \\
=& \sum_{k=1}^{K} \mathbf{KL}\left[q(\omega_k)||p(\omega_k)\right] \\
=& \sum_{k=1}^{K} \left[\sum_{j=1}^{J} \mathbf{E}_q \log q(\omega_{k,j}) - \sum_{j=1}^{J} \mathbf{E}_q \log p(\omega_{k,j}) \right]
\end{aligned} \tag{5.21}$$

5.3.3 Architecture

The encoding process consists of three neural networks:

- The $level_1$ label information $z_{i,1}$ generated model

  $q(z_{i,1}|y_i) = \mathcal{C}(\mu_{z,1}(y_i, \nu), \sigma_{z,1}(y_i, \nu))$.

- The $level_2$ label information $z_{i,2}$ generated model

  $q(z_{i,2}|y_i) = \mathcal{C}(\mu_{z,2}(y_i, \xi), \sigma_{z,2}(y_i, \xi))$.

- The hidden variable x generated model

  $q(x_i|y_i) = \mathcal{N}(\mu_x(y_i, \delta), \sigma_x(y_i, \delta))$.

where $\mathcal{C}(\cdot)$ is Gumbel-softmax[50].

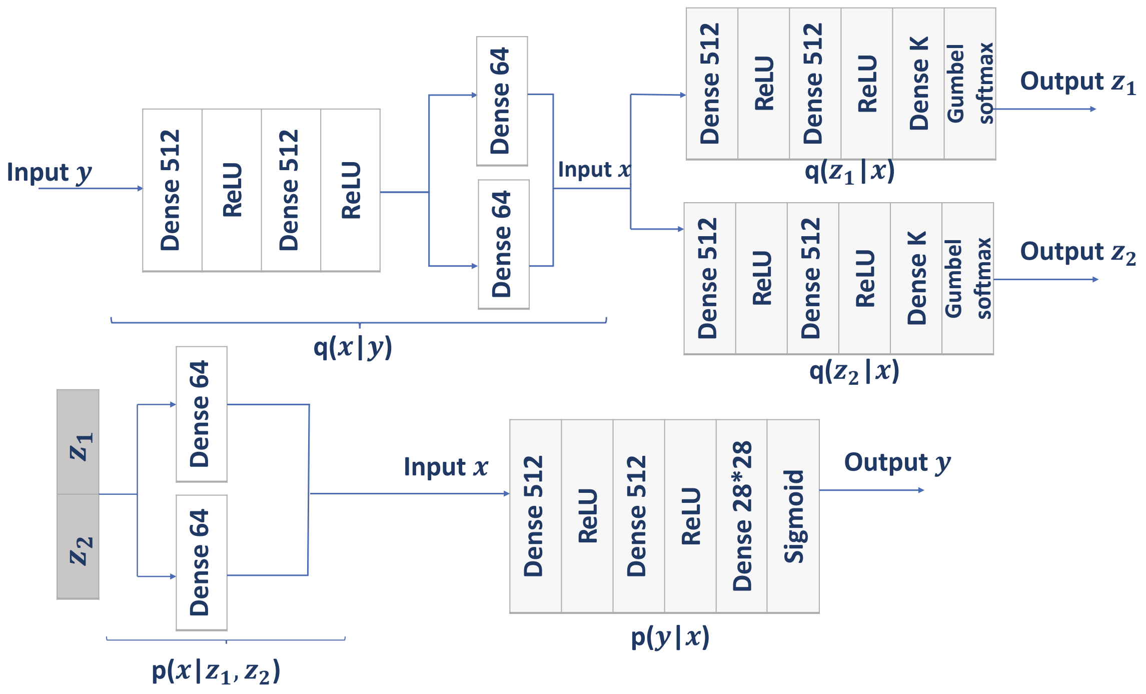


Figure 5.3 : The neural network architecture of I^{2}GMVAE.

The decoding process consists of two neural networks:

- The sample y reconstructed model

  $p(y_i|x_i) = \mathcal{N}(\mu_y(x_i, \phi), \sigma_y(x_i, \phi))$.

- The hidden variable x reconstructed model

  $p(x_i|z_i) = \mathcal{N}(\mu_x(z_{i,1}, z_{i,2}, \theta), \sigma_x(z_{i,1}, z_{i,2}, \theta))$.

These neural networks architecture is showed in Figure 5.3.

5.3.4 Training and Testing

We use six loss terms in the training stage, which are $\mathcal{L}_{rec}$, $\mathcal{L}_{gaus}$, $\mathcal{L}_{categ1}$, $\mathcal{L}_{categ2}$, $\mathcal{L}_{stick-breaking1}$ and $\mathcal{L}_{stick-breaking2}$, corresponding to the reconstruction loss term, x KL-divergence loss term, z_1 KL-divergence loss term, z_2 KL-divergence loss term, π KL-divergence loss term and ω KL-divergence loss term in ELOB respectively. Below we give the pseudo-code about how the updates are performed on I^{2}GMVAE model.

Algorithm 3 Training process of I^{2}GMM

```
**Required:** Traffic dataset $Y$
**Required:** $\alpha$ , $\gamma$, $K$, $J$
Feed forward $y$ to $q(x|y)$, $q(z_1|y)$ and $q(z_2|y)$ ,and get latent variable $x$, $z_1$ and $z_2$.
Feed forward latent variable $x$, $z_1$ and $z_2$ to $p(y|x)$ and $p(x|z)$, and get faked sample $y'$.
Calculate the whole loss
    $\mathcal{L}_{i^2gmvae} = \mathcal{L}_{rec} + \mathcal{L}_{gaus} + \mathcal{L}_{categ1} + \mathcal{L}_{categ2} + \mathcal{L}_{stick-breaking1} + \mathcal{L}_{stick-breaking2}$
Update parameter $\tau$, $\zeta$, encoder parameters $q(x|y)$, $q(z_1|y)$, $q(z_2|y)$and decoder parameters $p(y|x)$, $p(x|z_1, z_2)$ with $\mathcal{L}_{i^2gmvae}$.
```

5.4 Experiments and Results

5.4.1 Dataset and Evaluation Metrics

We evaluate our models on a wireless YouTube traffic dataset [65]. This dataset contains 10 YouTube videos. Each category has 300 samples.

We compare our models with state of art GMVAE of the same target of our models from clustering and generation aspects. We adopt three popular clustering metrics Unsupervised Clustering Accuracy (ACC), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and one generation metric bits/dim. ACC is an unsupervised form of classification accuracy. It uses a mapping function to map between predicted labels and ground truth labels. This mapping is necessary because unsupervised algorithms may use a different label than the ground truth to represent the same cluster. NMI measures the distance between cluster assigned labels and ground truth labels by measuring the mutual information between cluster assigned label and ground truth labels. It is normalized by the mean entropy of ground labels and cluster assigned labels. ARI is a modified version of the Rand index. The Rand index computes a similarity measure between two clusters by computing all pairs of samples and counts assigned in the same or different clusters in the predicted and true clusters. Bits/dim is a common measurement for sample generation. It is the total discrete log-likelihood is normalized by the dimensionality of the samples (e.g., $500\times 1 = 500$ for our dataset) [109].

5.4.2 Implementation Details

We implement our model with Pytorch [86]. The batch size is set as 64. We use the Adam optimizer, and the learning rate is 0.001. We use binary cross-entropy to compute the reconstruction loss. The initial temperature used in Gumbel-softmax is set as 1. And we decay Gumbel temperature every epoch. The temperature decay rate and minimum temperature are 0.0138 and 0.5. For IGMVAE, the max cluster number K is set as 500. The initial α is set as 1. For I^{2}GMVAE, the max cluster number K and J are set as 10 and 30. The initial α and γ are set as 1. The total training number is 100.

5.4.3 Results

We evaluate 1) whether our models learn a meaningful category discriminable latent space and can generate samples by category; 2) whether our models can do incremental learning when new categories data are added in training; 3) and we compare our two models with the state of art GMVAE [55].

IGMVAE Performance

In the first experiment, we compare our IGMVAE model with GMVAE. For GMVAE, we need manually set the cluster number. In this experiment, we set it 5, 10, 25, 50, and 100 to observe how cluster number affects the performance. For our model, the cluster number is learned by itself. As Figure 5.4 shows, the data is classified into other clusters rather than the top 18 high probability clusters with

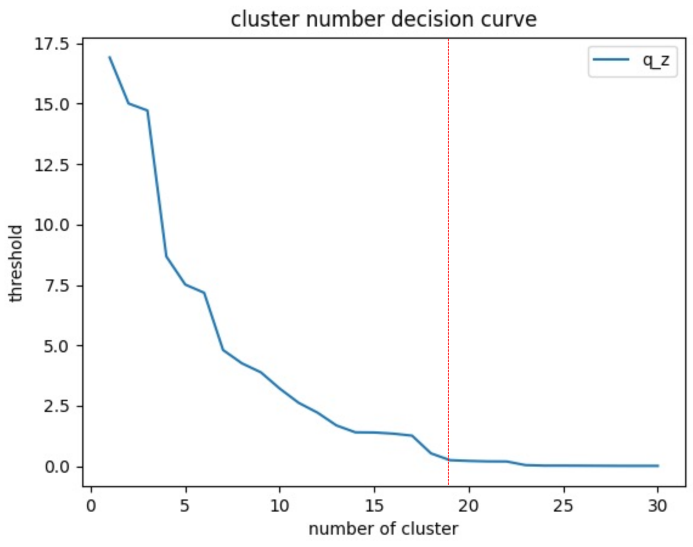


Figure 5.4 : Cluster number decision curve of IGMVAE on YouTube.

Table 5.1 : IGMVAE and I^{2}GMVAE results (on YouTube)

	ACC ↑	**NMI** ↑	**ARI** ↑	**bits/dim** ↓
GMVAE-5	0.435	0.514	0.372	15.10
GMVAE-10	0.691	0.689	0.578	15.10
GMVAE-25	0.499	0.604	0.388	15.48
GMVAE-100	0.510	0.612	0.399	15.23
IGMVAE(our)	0.588	0.679	0.527	14.94
I^{2}GMVAE(our)	0.882	0.897	0.816	15.09

a very small probability (close to zero), so the model automatically determines the cluster number is 18. The performance are detailed in Table 5.1. We observed that our method can achieve 0.59 cluster accuracy, 0.68 ARI score, 0.53 NMI score, and 14.94 bits/dim.

To better understand that our models realize sample generation. We visualize the generative samples of every cluster generated by generative network $q(y|x)$. In Figure 5.5, we can observe that our model can generate similar samples of the same cluster.

Furthermore, as we introduced in 5.2, IGMM has the ability to identify new categories, that is, the data that does not belong to the original training dataset

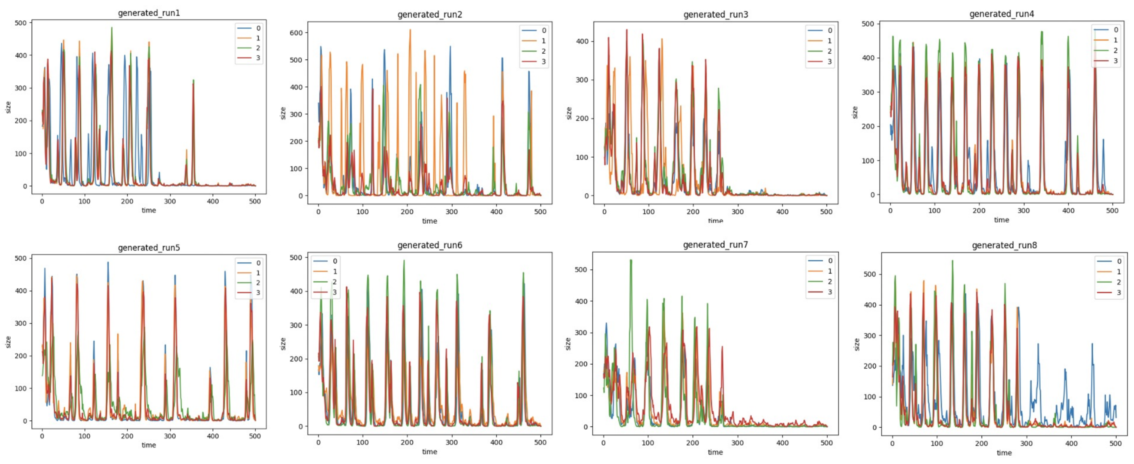


Figure 5.5 : Generated YouTube videos I/O graph of IGMVAE.

can be assigned to new clusters. So that we did an experiment called incremental experiment to test the adaptability of our model to the new categories data. In this experiment, we separate the dataset into two parts. One is base dataset(video 0 to 6), another is incremental dataset(video 7 to 9). This experiment consists of two stages. In the first stage, we feed the base dataset into the model, and the cluster number is learned by itself. In the second stage, we add the incremental dataset and the new cluster number is learned. The learned number of clusters of these stages is shown in Figure 5.6. We observed that when the incremental dataset was added, the cluster number changed from 12 to 16, which means the model has the ability to learn new clusters.

I*^{2}*GMVAE Performance

The model automatically determines the cluster number is 20. The performance of our I^{2}GMVAE on YouTube video dataset is detailed in Table 5.1. We observed that our method can achieve 0.88 cluster accuracy, 0.90 ARI score, 0.81 NMI score, and 15.09 bits/dim.

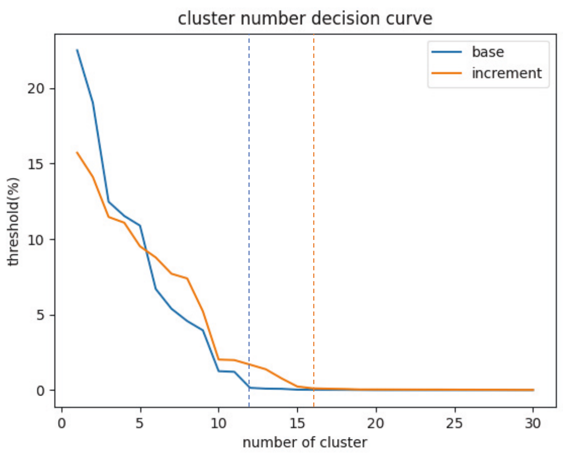
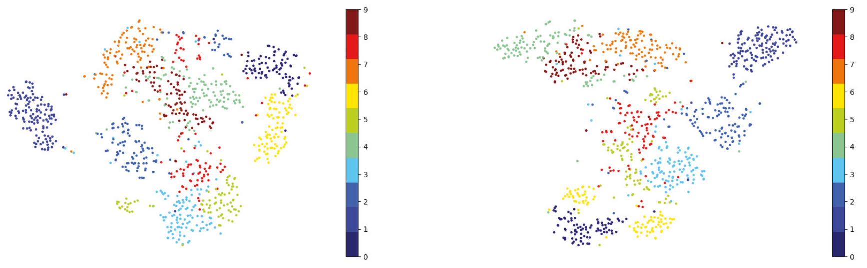
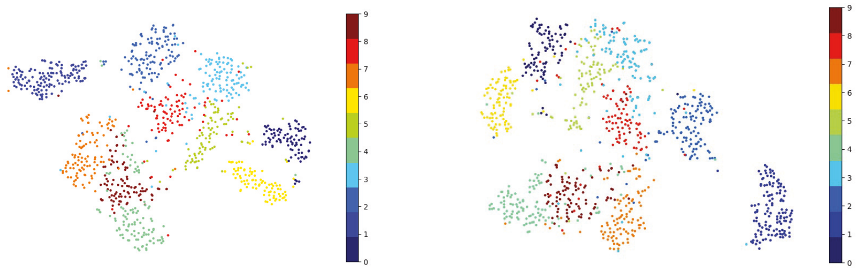


Figure 5.6 : Cluster number decision curve of IGMVAE for the incremental experiment (The blue color line is based on video 0 to 6. The orange color line is based on video 0 to 9).



(a) GMVAE latent variables with cluster number 5

(b) GMVAE latent variables with cluster number 10



(c) GMVAE latent variables with cluster number 50

(d) GMVAE latent variables

Figure 5.7 : Visualisation of the latent variables on YouTube: (a) GMVAE learns the latent variables with cluster number 5. (b) GMVAE learns the latent variables with cluster number 10. (C) GMVAE learns the latent variables with cluster number 25. (d) the latent variables of IGMVAE(our).

5.4.4 Result Analysis

In order to better visualize the effect of clustering, we map the latent variables to 2-dimensional data space. As shown in Figure 5.7, we can observe that the clustering ability of GMVAE is variate and unstable with different the number of clusters we are given. For our model, the cluster number is learned by itself and the clustering effect is stable and better.

And from Table 5.1, we can also see that the clustering performance of GMVAE is heavily dependent on the cluster number we set in training. When the given number of clusters is the same as the ground truth (10 for the YouTube video dataset), the performance is the best. The bigger the gap between the given cluster number and the ground truth, the worse the performance. For our two models, the clustering performances are slightly worse than GMVAE when the given cluster number is the ground truth, but better than it in other cases. Besides, the generative effect of our two models remains the same as GMVAE.

5.5 Summary

In this chapter, we first proposed a novel generative model, namely IGMVAE, which is a variant of VAE with IGMM priors. We showed that this model has the ability to generate samples for a specific cluster. And it has a good clustering ability without the cluster number given. Besides, we showed that our model has the ability to adopt new categories samples and learn the new clusters.

We next extended the idea of using a nonparametric Bayesian model as prior distribution of latent variables and proposed another novel generative model, namely I^{2}GMVAE, which is a two-level IGMVAE. We can understand it as a two-level of clustering. First, we do a coarse-grained clustering of the dataset, and then do a fine-grained clustering for each upper-level cluster. We showed that this method can generate data hierarchically and achieve a good performance.

We evaluate our two models on the traffic dataset compared with GMVAE. The results show that our models achieve better clustering performance and remain at the same level of generative effect.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Network traffic classification is an important work in modern network management and security systems. Today, with the explosion of network traffic and HTTPS has become the norm for many forms of communication over the Internet, traditional traffic classification methods(e.g. port-based and payload-based methods) have some practical problems, such as dynamic ports and encryption applications. So, deep learning technology based on traffic statistical characteristics gains more attention.

This thesis has conducted a study on deep learning based Wifi traffic classification. We proposed a series of models corresponding to this traffic content classification and thus formed three **Research Objective**: ① Classifying encrypted WiFi videos using deep learning models; ② Classifying encrypted WiFi traffic using a hierarchical classifier; ③ Generating samples by category using Bayesian nonparametric based autoencoders. The proposed machine learning models for the three objectives are discussed in Chapter 3-5.

In chapter 3, we demonstrate the possibility of making predictions from encrypted WiFi traffic by building deep learning based classifiers that are able to identify specific videos. For video traffic classification, we proposed three novel neural network models and show that our models are able to achieve around 97% accuracy in identifying videos from a closed set of 10 videos purely based on passive measurements collected at the WiFi layer.

In chapter 4, not just limited to video content, there are different types of traffic like audio and text, and there are many content providers that need to be classified. So that we present a novel hierarchical traffic classifier that is able to make coarse-grained and fine-grained predictions about encrypted traffic flow by leveraging weight sharing features in convolutional neural networks. Besides, we demonstrate our model's potential for classifying previously unseen content to its corresponding traffic type and content provider.

In chapter 5, we point out the importance of automatically generating traffic data in the network traffic domain. And we proposed two novel generative models, namely IGMVAE and I^{2}GMVAE. Specially, in our IGMVAE model, we use IGMM as the prior distribution of latent variables. In our I^{2}GMVAE model, we use IGMM as the prior distribution of latent variables. We show that our two models are able to generate samples by categories and have a good generative performance.

6.2 Future Work

The future research can be extended in but not limited to the following aspects:

- Our experiments are based on one assumption. We use the YouTube Red member account to avoid advertisements when collecting data. However, in actual scenarios, we cannot guarantee that there are no advertisements at the beginning or middle of the videos. In future work, we aim to develop methods to identify and preprocess advertising data.

- There is another assumption during data collection. That is, the starting point of the video is supposed to be known. Although it is essential to estimate and identify the video start point in a real scenario, to simplify the data collection, we ignored this procedure. In future work, we aim to address this issue, such as using traffic type classification techniques and increasing data capture time.

- Given that the classification accuracy over Spotify and Xiami is affected by the single-peak shape of their traffic, we plan to try to improve the performance by adding more dedicated pre-training data or exploring peak detection algorithms.

- In practice, it is difficult to create a classifier for all existing traffic classes, as well as keep up to date with any new content providers. How to solve the open-set problem, such as the traffic does not appear in the dataset.
- How to fine-tune the learned classifier to the new traffic data with a small cost and efficiently use the newly collected data to update the previously trained model.
- We can combine our proposed generative model with our classifier, which allows generation and classification to be performed simultaneously.
- We can further explore the generation effect of the fusion of the BNP model and other generative models, such as GAN.
- How to design a lightweight and fast data preprocessing algorithm. In practice, we can ensure that a large amount of data can be processed quickly and target data can be quickly identified.

Bibliography

[1] "Internet asigned numbers authority (iana)," https://www.iana.org.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[3] G. Aceto, A. Dainotti, W. De Donato, and A. Pescapé, "Portload: taking the best of two worlds in traffic classification," in *2010 INFOCOM IEEE Conference on Computer Communications Workshops.* IEEE, 2010, pp. 1–5.

[4] R. Alshammari and A. N. Zincir-Heywood, "Investigating two different approaches for encrypted traffic classification," in *Sixth Annual Conference on Privacy, Security and Trust.* IEEE, 2008, pp. 156–166.

[5] ——, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *2009 IEEE symposium on computational intelligence for security and defense applications.* IEEE, 2009, pp. 1–8.

[6] ——, "How robust can a machine learning approach be for classifying encrypted voip?" *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 830–869, 2015.

[7] C. E. Antoniak, "Mixtures of dirichlet processes with applications to bayesian nonparametric problems," *The annals of statistics*, pp. 1152–1174, 1974.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning.* PMLR, 2017, pp. 214–223.

[9] D. J. Arndt and A. N. Zincir-Heywood, "A comparison of three machine learning techniques for encrypted network traffic analysis," in *2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA).* IEEE, 2011, pp. 107–114.

[10] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Transactions on neural networks*, vol. 18, no. 1, pp. 223–239, 2007.

[11] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *International Conference on Passive and Active Network Measurement.* Springer, 2007, pp. 165–175.

[12] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[14] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[15] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z.-H. Ling, "Encrypted network traffic classification using deep and parallel network-in-network models," *IEEE Access*, vol. 8, pp. 132 950–132 959, 2020.

[16] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," *arXiv preprint arXiv:1509.00519*, 2015.

[17] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo, "A survey on encrypted traffic classification," in *International Conference on Applications and Techniques in Information Security.* Springer, 2014, pp. 73–81.

[18] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *2010 IEEE Symposium on Security and Privacy.* IEEE, 2010, pp. 191–206.

[19] G. Cheng and Y. Hu, "Encrypted traffic identification based on n-gram entropy and cumulative sum test," in *Proceedings of the 13th International Conference on Future Internet Technologies*, 2018, pp. 1–6.

[20] J. Cheng, Y. Wu, E. Yuepeng, J. You, T. Li, H. Li, and J. Ge, "Matec: A lightweight neural network for online encrypted traffic classification," *Computer Networks*, vol. 199, p. 108472, 2021.

[21] C. Cimpanu, "Video streams leak what you're watching to attackers with over 95% accuracy," https://www.bleepingcomputer.com/news/security/video-streams-leak-what-youre-watching-to-attackers-with-over-95-percent-accuracy/, 2017.

[22] cisco, "Nbar2 or next generation nbar," https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/network-based-application-recognition-nbar/qa_c67-697963.html, 2019.

[23] CISCO, "VNI complete forecast highlights," https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf, 2021.

[24] K. C. Claffy, "Internet traffic characterization." 1995.

[25] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[26] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of internet chat

systems," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 51–64.

[27] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv preprint arXiv:1611.02648*, 2016.

[28] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[29] T. S. Ferguson, "A bayesian analysis of some nonparametric problems," *The annals of statistics*, pp. 209–230, 1973.

[30] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen, "Service usage classification with encrypted internet traffic in mobile messaging apps," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2851–2864, 2016.

[31] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets.* Springer, 1982, pp. 267–285.

[32] S. J. Gershman and D. M. Blei, "A tutorial on bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, 2012.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[34] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on*

artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[35] S. E. Gómez, L. Hernández-Callejo, B. C. Martínez, and A. J. Sánchez-Esguevillas, "Exploratory study on class imbalance and solutions for network traffic classification," *Neurocomputing*, vol. 343, pp. 100–119, 2019.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[37] Google, "Video Streams Leak What You're Watching to Attackers With Over 95% Accuracy," https://www.bleepingcomputer.com/news/security/video-streams-leak-what-youre-watching-to-attackers-with-over-95-percent-accuracy/, 2018.

[38] ——, "HTTPS encryption on the web," https://transparencyreport.google.com/https/overview?hl=en&load_os_region=chrome-usage:1;series:page-load;groupby:os&lu=load_os_region, 2022, online; accessed 12-6-2022.

[39] L. Grimaudo, M. Mellia, and E. Baralis, "Hierarchical learning for fine grained internet traffic classification," in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC).* IEEE, 2012, pp. 463–468.

[40] C. Gu, S. Zhang, and Y. Sun, "Realtime encrypted traffic identification using machine learning." *J. Softw.*, vol. 6, no. 6, pp. 1009–1016, 2011.

[41] J. Gu, J. Wang, Z. Yu, and K. Shen, "Walls have ears: Traffic-based side-channel attack in video streaming," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications.* IEEE, 2018, pp. 1538–1546.

[42] Y. Guo, G. Xiong, Z. Li, J. Shi, M. Cui, and G. Gou, "Ta-gan: Gan based traffic augmentation for imbalanced network traffic classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[43] R. Hasibi, M. Shokri, and M. Dehghan, "Augmentation scheme for dealing with imbalanced network traffic classification using deep learning," *arXiv preprint arXiv:1901.00204*, 2019.

[44] G. He, B. Xu, L. Zhang, and H. Zhu, "Mobile app identification for encrypted network flows by traffic correlation," *International Journal of Distributed Sensor Networks*, vol. 14, no. 12, p. 1550147718817292, 2018.

[45] Z. Hejun and Z. Liehuang, "Encrypted network behaviors identification based on dynamic time warping and k-nearest neighbor," *Cluster Computing*, vol. 22, no. 2, pp. 2571–2580, 2019.

[46] M.-h. HONG, R.-t. GU, H.-x. WANG, Y.-m. SUN, and Y.-f. JI, "Identifying online traffic based on property of tcp flow," *The Journal of China Universities of Posts and Telecommunications*, vol. 16, no. 3, pp. 84–88, 2009.

[47] J.-T. Huang, J. Li, and Y. Gong, "An analysis of convolutional neural networks for speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4989–4993.

[48] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.

[49] H. A. H. Ibrahim, O. R. A. Al Zuobi, M. A. Al-Namari, G. MohamedAli, and A. A. A. Abdalla, "Internet traffic classification using machine learning approach: Datasets validation issues," in *2016 Conference of Basic Sciences and Engineering Studies (SGCAC)*. IEEE, 2016, pp. 158–166.

[50] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[51] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of p2p traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 121–134.

[52] C. Kattadige, K. N. Choi, A. Wijesinghe, A. Nama, K. Thilakarathna, S. Seneviratne, and G. Jourjon, "Seta++: Real-time scalable encrypted traffic analytics in multi-gbps networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3244–3259, 2021.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[55] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," *Advances in neural information processing systems*, vol. 27, 2014.

[56] K. Kohls, D. Rupprecht, T. Holz, and C. Pöpper, "Lost traffic encryption: fingerprinting lte/4g traffic on layer two," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks.* ACM, 2019, pp. 249–260.

[57] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," *Advances in neural information processing systems*, vol. 28, 2015.

[58] 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.

[59] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, p. 20, 2015.

[60] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, 2009.

[61] S. Leroux, S. Bohez, P.-J. Maenhaut, N. Meheus, P. Simoens, and B. Dhoedt, "Fingerprinting encrypted network traffic types using machine learning," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–5.

[62] Let's Encrypt, "Let's Encrypt Stats," https://letsencrypt.org/stats/, 2022, online; accessed 12-6-2022.

[63] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of wi-fi traffic," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1033–1047, 2018.

[64] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Computer Networks*, vol. 53, no. 6, pp. 790–809, 2009.

[65] Y. Li, Y. Huang, R. Xu, S. Seneviratne, K. Thilakarathna, A. Cheng, D. Webb, and G. Jourjon, "Deep content: Unveiling video streaming content from encrypted wifi traffic," in *17th Int. Symp. on Network Computing and Applications*. IEEE, 2018, pp. 1–8.

[66] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, "Effective and real-time in-app activity analysis in encrypted internet traffic streams," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 335–344.

[67] Y. Liu, J. Chen, P. Chang, and X. Yun, "A novel algorithm for encrypted traffic classification based on sliding window of flow's first n packets," in *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)*, 2017, pp. 463–470.

[68] Z. Liu, J. Wang, and Z. Liang, "Catgan: Category-aware generative adversarial networks with hierarchical evolutionary learning for category text generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8425–8432.

[69] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[70] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Auxiliary deep generative models," in *International conference on machine learning*. PMLR, 2016, pp. 1445–1453.

[71] A. Madhukar and C. Williamson, "A longitudinal study of p2p traffic classification," in *14th IEEE international symposium on modeling, analysis, and simulation*. IEEE, 2006, pp. 179–188.

[72] E. Mahdavi, A. Fanian, and H. Hassannejad, "Encrypted traffic classification using statistical features." *ISeCure*, vol. 10, no. 1, 2018.

[73] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[74] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[75] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *International workshop on passive and active network measurement.* Springer, 2005, pp. 41–54.

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

[77] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy, "The {CoralReef} software suite as a tool for system and network administrators," in *15th Systems Administration Conference (LISA 2001)*, 2001.

[78] T. T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks.* IEEE, 2006, pp. 369–376.

[79] B. Niemczyk and P. Rao, "Identification over encrypted channels," *BlackHat USA*, 2014.

[80] T. J. O'Shea, S. Hitefield, and J. Corgan, "End-to-end radio traffic sequence recognition with recurrent neural networks," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP).* IEEE, 2016, pp. 277–281.

[81] J. Palmer, K. Kreutz-Delgado, B. Rao, and D. Wipf, "Variational em algorithms for non-gaussian latent variable models," *Advances in neural information processing systems*, vol. 18, 2005.

[82] A. Panchenko, “Website fingerprinting in onion routing based anonymization networks,” in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society.* ACM, 2011, pp. 103–114.

[83] ——, “Website fingerprinting at internet scale.” in *NDSS*, 2016.

[84] B. Park, Y. Won, J. Chung, M.-s. Kim, and J. W.-K. Hong, “Fine-grained traffic classification based on functional separation,” *International Journal of Network Management*, vol. 23, no. 5, pp. 350–381, 2013.

[85] K. Park and H. Kim, “Encryption is not enough: Inferring user activities on kakaotalk with traffic analysis,” in *International Workshop on Information Security Applications.* Springer, 2015, pp. 254–265.

[86] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.

[87] V. Paxson, “Empirically derived analytic models of wide-area tcp connections,” *IEEE/ACM transactions on Networking*, vol. 2, no. 4, pp. 316–336, 1994.

[88] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.

[89] C. Rasmussen, “The infinite gaussian mixture model,” *Advances in neural information processing systems*, vol. 12, 1999.

[90] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, “Studying tls usage in android apps,” in *Proceedings*

of the 13th International Conference on emerging Networking EXperiments and Technologies, 2017, pp. 350–362.

[91] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate dash videos streamed over encrypted 802.11n connections," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2016, pp. 1107–1112.

[92] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy.* ACM, 2017, pp. 361–368.

[93] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," *arXiv preprint arXiv:1708.06376*, 2017.

[94] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for lvcsr," in *2013 IEEE workshop on automatic speech recognition and understanding.* IEEE, 2013, pp. 315–320.

[95] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "A review on machine learning–based approaches for internet traffic classification," *Annals of Telecommunications*, vol. 75, no. 11, pp. 673–710, 2020.

[96] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[97] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th USENIX Security Symposium*, 2017, pp. 1357–1374.

[98] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 512–521.

[99] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 137–150.

[100] J. Sethuraman, "A constructive definition of dirichlet priors," *Statistica sinica*, pp. 639–650, 1994.

[101] T. Seyed Tabatabaei, M. Adel, F. Karray, and M. Kamel, "Machine learning-based classification of encrypted internet traffic," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition.* Springer, 2012, pp. 578–592.

[102] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[103] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[104] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*, vol. 2. IEEE, 1999, pp. 246–252.

[105] M. Struwe and M. Struwe, *Variational methods.* Springer, 2000, vol. 991.

[106] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P).* IEEE, 2016, pp. 439–454.

[107] ——, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

[108] A. Tongaonkar, R. Keralapura, and A. Nucci, "Challenges in network application identification." in *LEET*, 2012.

[109] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International conference on machine learning.* PMLR, 2016, pp. 1747–1756.

[110] M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[111] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[112] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I know what you did on your smartphone: Inferring app usage over encrypted data traffic," in *2015 IEEE conference on communications and network security (CNS).* IEEE, 2015, pp. 433–441.

[113] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society.* ACM, 2013, pp. 201–212.

[114] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2017, pp. 43–48.

[115] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose, "Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks," in *2011 IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 3–18.

[116] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 35–49.

[117] ——, "Uncovering spoken phrases in encrypted voice over ip conversations," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 4, p. 35, 2010.

[118] Q. Xu, Y. Liao, S. Miskovic, Z. M. Mao, M. Baldi, A. Nucci, and T. Andrews, "Automatic generation of mobile app signatures from traffic observations," in *2015 IEEE conference on computer communications (INFOCOM)*. IEEE, 2015, pp. 1481–1489.

[119] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2740–2748.

[120] H. Z. Yerebakan, B. Rajwa, and M. Dundar, "The infinite mixture of infinite gaussian mixtures," *Advances in neural information processing systems*, vol. 27, 2014.

[121] J. Yu, H. Lee, Y. Im, M.-S. Kim, and D. Park, “Real-time classification of internet application traffic using a hierarchical multi-class svm.” *KSII Transactions on Internet & Information Systems*, vol. 4, no. 5, 2010.

[122] L. Yu and H. Liu, “Feature selection for high-dimensional data: A fast correlation-based filter solution,” in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856–863.

[123] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent neural network regularization,” *arXiv preprint arXiv:1409.2329*, 2014.

[124] Y. Zhu and H. Fu, “Traffic analysis attacks on skype voip calls,” *Computer Communications*, vol. 34, no. 10, pp. 1202–1212, 2011.

[125] D. Zuev and A. W. Moore, “Traffic classification using a statistical approach,” in *International workshop on passive and active network measurement.* Springer, 2005, pp. 321–324.