MULTIMEDIA TRAFFIC CLASSIFICATION BASED ON DISCRETE TIME MARKOV CHAINS

by
OGUZ KAAN KOKSAL

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Electronics Engineering

Sabancı University
January 2023

MULTIMEDIA TRAFFIC CLASSIFICATION BASED ON DISCRETE TIME MARKOV CHAINS

Approved by:

Assist. Prof. HÜSEYİN ÖZKAN .....
(Thesis Advisor)

Prof. ÖZGÜR GÜRBÜZ .....
(Thesis Co-advisor)

Prof. AYŞE BERRİN YANIKOĞLU .....

Prof. ÖZGÜR ERÇETİN .....

Assist. Prof. H. BİRKAN YILMAZ .....

Date of Approval: January 26, 2023

ABSTRACT

MULTIMEDIA TRAFFIC CLASSIFICATION BASED ON DISCRETE TIME MARKOV CHAINS

OĞUZ KAAN KÖKSAL

Electronics Engineering M.Sc. THESIS, January 2023

Thesis Supervisor: Assist. Prof. Dr. Hüseyin Özkan
Co-advisor: Prof. Dr. Özgür Gürbüz

Keywords: Wi-fi Networks, Multimedia, Markov Chains, Machine learning, ECOC, Classification

Traffic prioritization has recently become critical for home Wi-Fi networks due to the increased number of connected devices and wide variety of applications. While some of these applications are delay sensitive, some have high throughput requirements. Therefore, managing different traffic types adaptively with regard to their requirements is crucial for a better Quality of Experience (QoE). Traffic type classification can be used to that end for detecting the specific requirements and enhance the Quality of Services (QoS). In the scope of the thesis, we propose to model the multimedia traffic flow as a stochastic discrete-time Markov chain (DTMC) in order to take into account the strong sequentiality (i.e. the dependencies across the data instances) in the traffic flow observations. Within that approach four novel data-driven classification schemes are presented. The first one is k Nearest Markov Components (kNMC) which relies on a Markov modeling of bit-rate. kNMC considers a mixture of Markov components and classifies by using a log-likelihood-based distance between train and test instances. The second classifier is kNMC-3D which applies the same technique with kNMC but focuses on the number of packets and inter-arrival times besides bit-rate. The other two classifiers exploit Error Correcting Output Codes (ECOC) for solving the multiclass problem with multiple binary kNMC-3D classifiers. The third classification scheme namely Confusion Based ECOC (CB-ECOC) proposes a custom-designed ECOC matrix which addresses the errors of kNMC-3D. The fourth classifier named as 2-Level ECOC, adds another classification level to CB-ECOC

for resolving the Skype identification issue of CB-ECOC. Considering multimedia data from popular applications such as Youtube, Netflix, Skype, Whatsapp, and Spotify from our introduced dataset, average traffic type classification accuracies are obtained up to 96.15% at the application level. Considering the given applications in different traffic categories, such as, Video on Demand (VOD), Sharing and Media Screening (S&MS), Video Live Streaming (VLS), and Teleconferencing (TC), average classification accuracies up to 97.75% are reached at the category level. The presented classifiers are also evaluated with the benchmark dataset from the literature and average classification accuracies are observed up to 97.75% at the application level, and up to 99.59% at the category level. In our extensive experiments, we observed that the introduced classifiers are highly accurate as compared to prominent competitors such as Support Vector Machines (SVM), Random Forest (RF), autoencoders and problem-independent ECOC models, e.g, One Versus One (OVO) and One Versus All (OVA).

ÖZET

AYRIK ZAMANLI MARKOV ZINCIRLERINI BAZ ALAN MULTIMEDYA TRAFIK SINIFLANDIRMASI

OĞUZ KAAN KÖKSAL

ELEKTRONİK MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, Ocak 2023

Tez Danışmanı: Assist. Prof. Dr. Hüseyin Özkan
İkinci Danışman: Prof. Dr. Özgür Gürbüz

Anahtar Kelimeler: Wi-fi Ağları, Multimedia, Markov Zincileri, Makina Öğrenmesi, Hata Düzeltme Çıktı Kodları, Sınıflandırma

Son yıllarda evlerde sayıları artan ağa bağlı cihazların ve kullanılan Skype ,Netflix gibi aplikasyonların yaygınlaşması nedeniyle, Wi-fi ağlarında trafiğin önceliklendirilmesi büyük önem kazandı. Bu uygulamaların bazıları gecikmeye duyarlı iken bazıları ise yüksek veri iletim hızına gerek duymaktadır. Bu yüzden farklı uygulamalardan gelen trafik tiplerini yönetirken trafiklerin tiplerine özel gereksinimleri dikkate almak kullanıcı tecrübesi açısından önemlidir. Dolayısıyla trafik tiplerinin sınıflandırılması, bu gereksinimlerin saptanması ve servis kalitesinin (QoS) artırılması için kullanılabilir. Bu tez kapsamında biz ardı ardına gelen trafik paketlerinin arasındaki güçlü bağlantıları dikkate almak için multimedya trafik akışlarının stokastik ayrık zamanlı Markov zincirleri olarak modellenmesini önerdik. Bu yaklaşımla çalışan dört yeni veri tabanlı sınıflandırma şeması sunduk. Birinci sınıflandırıcı olan ve bit hızının Markov modellemesine dayanan “En Yakın k Markov Bileşeni Sınıflandırıcısı" (kNMC), Markov bileşenlerinin karışımını dikkate alır ve sınıflandırmada eğitim ve test verilerinin arasındaki olabilirlik oranının logaritmasını baz alan bir uzaklık ölçümü kullanır. İkinci sınıflandırıcı olan “Üç Boyutlu En Yakın k Markov Bileşeni Sınıflandırıcısı" (kNMC-3D) birinci sınıflandırıcı ile aynı yöntemleri kullanır ama bit hızının yanında paket sayısı ve peş peşe gelen paketler arası sürelerin ortalamasını da modeline dahil eder. Diğer iki sınıflandırıcı tanıtılan çok sınıflı problemi çözmek için Hata Düzeltme Çıktı Kodlarını (ECOC) çok sayıda ikili kNMC-3D'yle beraber kullanır. Hata Matrisi Odaklı-ECOC (CB-

ECOC) olarak çağrılan üçüncü sınıflandırıcı, kNMC-3D'nin hatalarını düzeltmek için özel olarak tasarlanmış bir ECOC matrisi sunar. Dördüncü sınıflandırıcı olan 2-Seviye-ECOC, üçüncü sınıflandırıcının Skype paketlerinin belirlenmesi konusundaki hatalarını çözmek amacıyla ilave bir sınıflandırma seviyesi ekler. Youtube, Netflix, Skype, Whatsapp, Spotify gibi uygulamaların verilerini içeren bir veri seti topladık. Yaptığımız deneylerde %96.15'e kadar doğruluk oranları elde edilmiştir. Ayrıca bahsedilen aplikasyonların Talep Üzerine Video (VOD), Paylaşım ve Medya Akışı (S&MS), Canlı Video Akışı (VLS) ve Telekonferans (TC) gibi trafik categorileri de sınıflandırılmış ve dört sınıflandırıcı ile %97.75 doğruluk değerlerine ulaşılmıştır. Tanıtılan sınıflandırıcılar ayrıca litaretürden bir kıyaslama veri kümesi ile değerlendirilmiş ve aplikasyon seviyesinde %97.75 doğruluk değeri, kategori seviyesinde %99.59 doğruluk seviyei gözlemlenmiştir. Tanıtılan sınıflandırıcıların; Destek Vektör Makinaları (SVM), Rastgele Orman (RF), Oto Kodlayıcılar (AE) ve problemden bağımsız ECOC metotları olan Bir vs Bir (OVO), Bir vs Hepsi (OVA) gibi rakip algoritmalar ile karşılaştırıldığında daha yüksek başarı oranı ile çalıştığı gözlemlenmiştir.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Internet is composed of different types of multimedia traffic and a significant portion belongs to video applications, such as video-on-demand and streaming video, each having its own characteristics and Quality of Service (QoS) requirements [1]. In order to provide the best end-to-end (ETE) user experience, network protocols and components should act according to the type of traffic while also considering the characteristics and QoS requirements. Although the traffic sources are well-aware of the type of traffic they generate, this information is often lost afterwards in the network due to the lack of support from the applications and policies of the autonomous systems forming Internet [2, 3]. Consequently, severe performance degradation occurs for especially traffic with stringent QoS requirements. Therefore, it is of paramount importance to be aware of the traffic type for any network component and at any time.

One such critical component is the wireless last-hop, for which one of the most widespread access technology is WiFi. Despite being the final step of a possibly long network path, if mishandled, the wireless last-hop can become a major bottleneck and introduce huge packet delays and jitter (i.e., delay variations) on the overall traffic [4]. The QoS amendment of WiFi (2005), namely IEEE 802.11e, offers four access categories (AC) for WiFi transmissions: video (AC_VI), voice (AC_VO), best effort (AC_BE), and background (AC_BK). [5]. By assigning appropriate ACs to the traffic flows, a prioritization mechanism can be realized in the wireless last-hop by changing the random access parameters (i.e., deference and backoff periods) for the CSMA/CA-based standard medium access control (MAC) layer, as in [6]. Alternatively, the newest generation of WiFi standard family, namely IEEE 802.11ax, offers Orthogonal Frequency Division Multiple Access (OFDMA)-based scheduled access, where an access point (AP) is capable of scheduling transmissions of individual stations (STAs) while considering their traffic characteristics and QoS requirements [7]. In both access schemes, the wireless channel is allocated with regards to the traffic types in order to achieve the desired quality of experience (QoE) and best user experience [8]. However, and importantly, both require the MAC layer

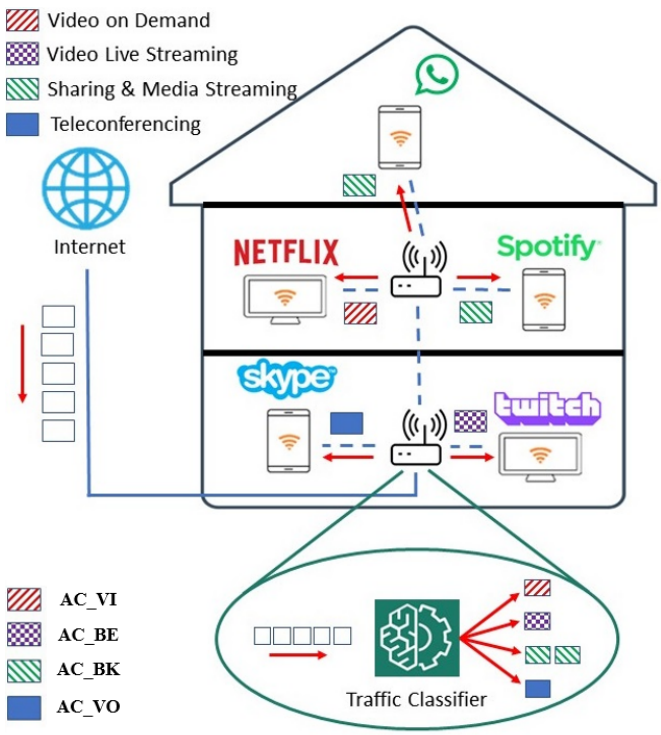

Figure 1.1 Envisioned home networking scenario with the proposed traffic classification to identify applications both category-wise and individually

of the WiFi interface to be aware of the type of the traffic that passes through, which requires reliable and accurate traffic classification schemes motivating the presented work.

In terms of traffic categorization, features that are obtained from the flow and rate signal—represented as a time series of instantaneous traffic rates. However, the considerable sequentiality (statistical dependency across data instances) in the traffic observations is a significant characteristic of the multimedia traffic categorization problem. Due to significant correlations, for example, a relatively high instantaneous rate is commonly followed by another high rate in the following time step, but one does not frequently notice abrupt changes between high and low rates in an uncorrelated fashion. On the other hand, the current methods often extract features from a specified window of traffic flow observations as a whole, such as the average inter-arrival periods of the packets. they disregard sequentiality. The biggest negative effect of that situation is information loss coming from consecutive packet relations.

To this end, in this thesis, DTMC-based four classification schemes are proposed for identifying various popular video and social media applications such as Youtube, Youtube-live, Whatsapp, Spotify, Netflix, Twitch and Skype in a network scenario, such as Figure 1.1. Considered applications can be grouped into different traffic

categories, namely, Video on Demand (VOD), Video Live Streaming (VLS), Sharing and Media Streaming (S&MS) and Teleconferencing (TC), which correspond to the aforementioned access categories of Wi-fi in Figure 1.1,

The first classifier of the thesis which is also the basis for others is k Nearest Markov Components(kNMC). The main idea behind kNMC is DTMC modeling of the aggregated number of bits in certain periods. After the aggregation process, whole chunk of number of bits data is quantized into some fixed levels that represent the states of the first-order DTMC. Within DTMC model, the local approach which considers the mixture of Markov components define kNMC. The second proposed classifier is kNMC-3D which is three-dimensional version of kNMC as its name implies. While kNMC use just number of bits info in the stochastic model, kNMC-3D utilizes, the number of packets and inter-arrival times besides the number of bits The other difference of kNMC-3D from kNMC is the usage of k-means as a quantizer. The other two classifiers are based on Error Correcting Output Codes (ECOC) classifiers which combine multiple binary classifiers to solve the given multiclass problem. The third classifier is the Confusion-Based ECOC Classifier (CB-ECOC) which uses multiple binary KNMC-3D's. CB-ECOC proposed a unique matrix that includes twelve sub-classifiers that focus on the points where multiclass kNMC-3D fails. The last classifier is the Two-Level ECOC Classifier (2L-ECOC). The main idea behind 2L-ECOC is that overall predictions are decided at the end of two-level classification. In the first level, CB-ECOC is operated and in the second level extra binary classifier focuses on the Skype identification problem of CB-ECOC. Adding an extra classification level is preferred rather than extending the ECOC matrix of CB-ECOC more because the solidity of the ECOC matrix could be affected in a bad way. In summary, all four proposed classification schemes are based on DTMC models of traffic flows. While the first classifier, kNMC, is a base for the other three classifiers, kNMC-3D, CB-ECOC, 2L-ECOC, improve kNMC with different approaches.

1.1 Contributions and Highlights

Main contributions and highlights of the thesis can be listed as follows.

- We propose to model the traffic flow features such as bitrate, number of packets and average inter-arrival times with a stochastic discrete-time Markov chain (DTMC) after quantization. This yields an observation/likelihood model as a

mixture of Markov components, which is experimentally highly effective.

- Based on the introduced DTMC modeling with only the bitrate feature, a novel local classifier namely kNMC (k nearest Markov components) is presented. By using three traffic features, bitrate, number of packets, and average inter-arrival times in the model, kNMC is extended to kNMC-Three Dimensional (kNMC-3D)

- Two novel error-correcting output codes (ECOC) based classifiers, which address the errors of kNMC-3D classifiers by observing confusion matrix, are proposed. First one named as Confusion-Based ECOC (CB-ECOC), mainly uses a special ECOC matrix designed with rule-based inference from the confusion of kNMC3D. The other classifier, namely two-level ECOC (2L-ECOC), adds another classification level for resolving Skype identification issues of CB-ECOC.

- In our experiments with the two datasets, the proposed kNMC classifier is observed to be highly accurate, achieving 85.71% at the application level and 89.11% at the category level with our dataset providing 78.61% accuracy at the application level and 98.24% accuracy at the category level with the benchmark dataset.

- Experiments with our dataset show that 84.98% accuracy at the application level and 91.13% accuracy at the category level are reached with kNMC-3D. Additionally, 90.73% accuracy at the application level and 99.04% accuracy at the category level are observed on the benchmark dataset.

- In addition, our results indicate the superiority of the proposed kNMC and kNMC-3D over the state-of-the-art methods such as SVM, random forest and autoencoder.

- With the proposed CB-ECOC classifier, on our dataset, 92.39% accuracy is achieved at the application level and 92.8% accuracy is observed at the category level. On the other hand with CB-ECOC on the benchmark dataset, accuracy values of 91.36% and 99.09% are observed at the application and category level, respectively.

- With the proposed 2L-ECOC classifier, on our dataset, 96.15% accuracy is achieved at the application level and 97.75% accuracy is observed at the category level. On the other hand with CB-ECOC on the benchmark dataset, accuracy values of 96.17% and 99.59% are observed at the application and category level, respectively.

- Extended experiments show that proposed ECOC-based classifiers provide better performance as compared to problem-independent ECOC schemes namely, One versus One (OVO) and One versus All (OVA).
- We emphasize that accurate multimedia traffic classification into applications and their categories is crucial, since then the underlying MAC level mechanisms (e.g., IEEE 802.11e or IEEE 802.11ax for WiFi) can be facilitated to ensure the required QoS. The presented approach can be used for this purpose successfully (as demonstrated in our performance evaluations) in not only WiFi but also other wireless last-hop alternatives as well as wired networks.

1.2 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2 a comprehensive summary of the related work is provided, comparing existing solutions with the proposed schemes. In Chapter 3, the DTMC modeling of network traffic and proposed classifiers that focus on DTMC modeling are presented. The details of ECOC-base classification are introduced in Chapter 4. In Chapter 5, experimental results with different datasets are provided, comparing proposed classifiers to prominent schemes from the literature. Chapter 6 presents our conclusions.

2. RELATED WORK

Previous work on traffic classification typically employs one of the four approaches: handling of packet tags, mapping packet address information, deep packet inspection (DPI), and analyzing packet flows with various machine learning techniques [9, 10, 11]. The ones of handling packet tags usually focus on the DiffServ Code Point (DSCP) tags within the IP packet headers. With the recent WebRTC protocol effort being supported by the IETF recently, the usage of DSCP tags has gained importance. In this method, the applications are expected to tag the packets at generation according to respective QoS classes [12, 13]. Then, the network components utilize DSCP tags in their traffic prioritization decisions. For the specific case of the wireless last-hop with WiFi, the DSCP tags are mapped into the 802.11e ACs [14, 15]. However, DSCP tags are set in only a small portion ($2-8.5\%$) of the overall Internet traffic [2, 3]. Studies show that often times, the DSCP tags are remarked and zeroed via the routers of the intermediate ASs along the path between the source and destination nodes [16, 17]. Therefore, usage of DSCP tags for a reliable method of traffic classification becomes very limited.

Classification by mapping packet address information is generally conducted by using the transport layer port addresses (i.e. the well-known port numbers) and the network layer IP addresses of the packets. Among the two, port addresses could be used to differentiate between packets belonging to different internet services based on the Internet Assigned Numbers Authority (IANA)-registered ports [18]. However, since many modern applications such as Netflix and YouTube use the same HTTP protocol at the application layer with the NAT and PAT mechanisms, the applicability of this method is restricted. Recently, services such as Microsoft Skype and Teams have offered new application protocol interfaces (APIs) for querying the IP and port address information of their servers [19]. By using these APIs, packet streams can be checked if they belong to such a service. While being quite useful for the services offering such APIs, the usefulness of this method is limited, as it does not apply to general traffic classification and it can only be used in the framework of certain services.

DPI methods analyze the contents of each packet to categorize according to the application specific payloads. These methods are generally powerful in classifying traffic at the packet level granularity with the cost of high computational complexity [20]. However, the utilization and effectiveness of DPI methods are generally deemed to be low due to i) the widespread usage of payload encryption in internet packets as well as ii) the privacy regulations limiting Internet Service Providers' ability to conduct such detailed level of inspection over packets without customer consent [9].

Machine learning based techniques, on the other hand, analyze traffic flows and generate various descriptive features like packet size, packet inter-arrival time, packet transmission, etc. Various classification schemes are then used with such features to recognize the traffic type [9]. In [21], random trees and C4.5 decision trees are used to differentiate Skype and non-Skype traffic. Video, audio and control flows composing a video streaming application are classified in [22] via support vector machines. Various background and multimedia applications are classified with k-nearest neighbor (kNN), J48, and random forests in [23]. A more recent work in [24] considers popular video streaming applications, namely Netflix and Youtube, employing aforementioned approaches. The traffic flow is chunked into small sub-flows in [25] and features of these sub-flows are used by the two naive Bayes and C4.5 decision trees based classifiers over gaming and VoIP traffic. This approach has been shown to provide decisions that are timely and also it does not impact the accuracy considerably, as opposed to the alternative of analyzing the entire flow at once without chunking. Similar to the sub-flow approach, the mobile encrypted application traffic flow is decomposed into the service bursts (SBs) in [26] and [27] with a pre-defined burst threshold. In [26], the bitflow features extracted from SBs are used by classifier fusion techniques which combine different base classifiers. In [27], various statistical features are generated from bursts and used by six classification approaches which are either support vector machines (SVM) or random forest based. Even though such classification approaches provide timeliness in both [26] and [27], the decomposition or burstification process requires manual inspection of the port and IP numbers. [28] combines three different known classifiers to into a new classifier named as Signature Static Port Classifier (SSPC) to classify online games. 91% average accuracy is achieved in two-stage classification, (one online and other one offline). In [29], the problem of port- based classifiers is addressed via a machine learning-based algorithm which uses inter-arrival time. In the specific case of a WiFi-based wireless last hop, MAC layer features (e.g., modulation and coding scheme or short ACK count) can be used for traffic classification [30]. In [31], a Markov chain of the message types of secure socket layer (SSL) traffic directly models the underlying stochastic process rather than indirectly extracting hand-

crafted features from the traffic. Then, the classification follows in a probabilistic framework.

A comprehensive analysis of deep learning (DL) approaches for multimedia traffic classification is provided in [32], where autoencoders, convolutional neural networks and recurrent neural networks (RNN) are compared. A hybrid DL model in [33] utilizes a stacked autoencoder followed by a softmax regression layer, where the classification performance is demonstrated based on the Moore dataset [34] that consists of 248 flow features (extracted from TCP flows of 10 different application categories: Bulk, Database, Interactive, Mail, Services, WWW, P2P, Attack, Games, Multimedia). The autoencoder learns from all available features whereas the regression layer realizes the classification with the learned features, achieving a classification performance higher than the alternative of support vector machines in their comparisons. An improvement over this method of [33] is achieved in [35] by a deep neural network with additional batch normalization and dropout layers. A different theme is presented in [36] where the goal is to investigate big data enabled and DL based classification of encrypted mobile traffic and, correspondingly, to provide an experimental setup for analysing the trade-offs among completion time, deployment costs, and classification performance.

Features that are derived from the flow and rate signal (as a time series of instantaneous traffic rates, cf. Fig. 3.2 for examples) tend to perform well in traffic classification. However, an important property of the multimedia traffic classification problem is the strong sequentiality (statistical dependency across data instances) in the traffic observations. For instance, a relatively high instantaneous rate is typically followed in the next time step by another high rate due to strong correlations, but one certainly does not frequently observe sudden switches between high and low rates in an uncorrelated fashion. On the other hand, the existing methods typically extract features, such as the average inter-arrival times of the packets, from a given window of traffic flow observations as a whole, which disregards the valuable sequence information. As another example, a traffic flow that steadily downloads and another one that sporadically downloads can have the same average, and therefore, can be mapped to the same feature by the average packet size which would suppress the discrimination power. In spite of the empirical evidence supporting the rate signal-derived features, it is mostly unclear how to precisely distill the rate signal and generate the best features while also considering the sequentiality.

For this reason, in this thesis, we opt to be agnostic and let the data decide what to look into by directly modelling the traffic flow and rate signal as a Markov model instead of a reduction through an indirect feature extraction. We emphasize that

any covariance-stationary process can be represented without losing information (Wold decomposition [37]) by a Markov model with infinite state space and a sufficiently large Markov order. Nevertheless, we consider a first order (for simplicity) discrete-time Markov model and a finite set of states via rate quantization which readily yields high performance in our experiments; but the generalization of our approach to any higher order and larger space of states is straightforward. Parameter estimations of our resulting Markov classification schemes can be recursively implemented computationally highly efficiently with negligible space requirements in real time. If our first order Markov approach was not powerful enough, then we would be increasing the order and the number of states which in turn would increase the parameter complexity perhaps intractably. In this case, an RNN would be more reasonable since it can cover high order statistical dependencies with tractable complexity thanks to its parameter sharing across layers. However, we confidently observe that a first order Markov is simpler than an RNN complexity-wise, and effective, for multimedia traffic classification.

There exists numerous examples of Markov models for time series analysis, in particular classification and anomaly detection. In [38], the mistakes of a first stage classifier for windows of a time series are corrected by using a hidden Markov model (HMM). In this example, the first stage does not account for the temporal information whereas the second stage trains an HMM where the Viterbi decoding provides improvements in accuracy as it finds the most likely sequence (but not the sequence of most likely individual states unlike the first stage). An HMM is proposed in [39] as a framework for anomaly detection in multi-variate time series data. The multivariate data is first transformed to a sequence of scalars for which two unknown underlying Markov states (normal or abnormal) are assumed. Then, Viterbi decoding after training an HMM recognizes the anomalies based on the sequentiality. For video identification, the sequence of foreground labels are modeled as Markov in [32], but not hidden as they can be directly identified by background subtraction, and then several Markov based features are offered for anomaly detection. Similarly, a Markov model with observable states is considered in [40] for anomaly detection, where the Markov states are obtained by a direct discretization of the observed amplitude values.

Our modeling of a continuous time stochastic process, the traffic flow rate signal in this thesis, in a first order and finite state Markov manner is inspired by the work [40], which -however- considers a completely different goal of online Neyman-Pearson anomaly detection with false positive rate controllability. In this sense, modelling the traffic flow rate signal enables us to fully exploit the rate information in a unique fashion, and greatly distinguishes our technique from the existing literature as well

as the work [31] which only considers a Markov sequence of message types and disregards the rate. A further uniqueness of our method is that we carefully fine-tune the Markov model (with extensive and proper cross-validations) to the multimedia traffic classification problem with the right complexity which is extremely important. Namely, the number of Markov states or quantization levels, Markov order, the frequency of sampling from the traffic flow and finally the boundaries of the quantization levels are all carefully investigated in our presented work. In addition, we do not require any tag information or any API lookup, and use only unencrypted parts of the packets to preserve user privacy. On the other hand, most ML based studies generate features based on the entire traffic flow and thus largely reduce the timeliness which is especially critical for delay-sensitive and live applications. In contrast, our presented technique is specifically designed to provide timeliness with limited latency.

3. TRAFFIC CLASSIFICATION BASED ON DISCRETE TIME MARKOV CHAINS

In this chapter, the DTMC modeling of the multimedia traffic data is explained in detail. First, a description and collection of our dataset are provided and our DTMC approach is presented via the analysis of network data, in particular the bitrate signal. This is followed by the construction and detailed discussion of the proposed Markov-based classifiers.

3.1 Dataset Description

On actual End to End connections with a Wi-Fi network as the final hop, we investigate a variety of multimedia applications. The network is set up as illustrated in Figure 3.1, with two WiFi STAs acting as clients executing the applications. STAs connect to the Internet through a home-network-grade WiFi AP, and the monitoring device captures packets using the Wireshark [41]. STAs replicate the behavior of an indoor WiFi user by moving around in a simple nomadic motion. In order to reduce the effect of specific traffic management, ETE connections are considered over two different Internet Service Providers (ISPs), i.e., some traffic flows are captured while the AP is connected through the first ISP, and the rest of the traffic flows are captured while the AP is connected through the second ISP.

We focus on a total of seven applications corresponding to four categories. Five of them are video-heavy applications in three categories as shown in Fig 3.1: Netflix and YouTube representing the VOD category, YouTube Live and Twitch forming the VLS category, and Skype representing the TC category. To increase the diversity of traffic flows and traffic data, we also consider two non-video multimedia applications Spotify and WhatsApp, and a fourth category S&MS. During the process of data collection, two user devices (clients), which are connected to the Internet via the AP

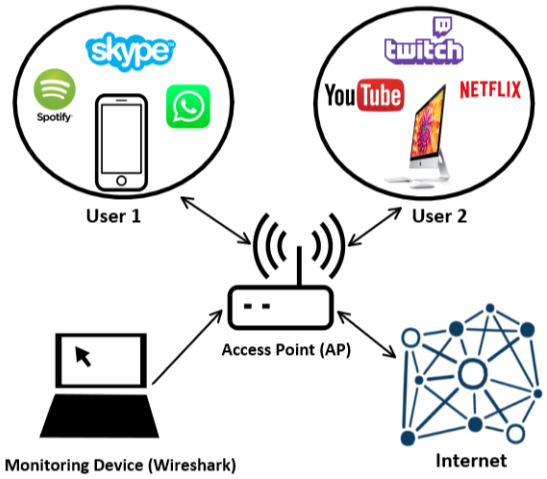

Figure 3.1 Data collection setup, multimedia categories and applications

in a WiFi network, run different applications as the Wireshark packet analyzer tool [41] on a third device sniffs and captures the downlink internet traffic that passes through the AP. Note that the Wireshark tool operates at the MAC layer and - by using the monitoring mode- captures each user's packets separately. For each of these seven applications, we capture 30 instances (per application) of downlink (DL) internet traffic streamed from the server via the AP to user as well as its uplink (UL) traffic. Each instance as a trace of the corresponding application is ten minutes long, which yields traffic flows of 300 minutes per application in total. In other words, only one specific application is run (for ten minutes) each time and the resulting instance, i.e., trace, is labeled with the name of the corresponding application for ground truth association. Also, there is a one-minute silent period between two runs to ensure that there is no residual background traffic in switching from one run to another. As a final step, we convert the captured packet based traffic data to flow based rate signals for each instance separately. During this conversion, a rate sample at time t is obtained as the total number of bits in packets between times $t-\Delta/2$ and $t+\Delta/2$ (sec). Here, Δ is the aggregation time for which the best choice is observed to be $1/50$ sec in our experiments. This constitutes the introduced dataset which we have made publicly available: $\{u_{i,t}, l(u_{i,t})\}_{i=1}^{N_u}$, where $l(u_{i,t})$ represents the corresponding application label, $N_u = 30 \times 7 = 210$ is the number of collected rate signals and each rate signal $u_{i,t}$ is of $T = 600 = 60 \times 10$ seconds. Several rate signal examples for various applications are given in Fig. 3.2. The feature analysis of the dataset is presented in Table 3.1.

Various packet-level and flow-level features are investigated for the introduced multimedia traffic dataset. These are three flow-level features, i) total traffic size, ii) mean traffic rate and iii) DL/UL ratio, as well as three packet-level features, i) the standard deviation of packet length, ii) mean inter-arrival time, and iii) standard

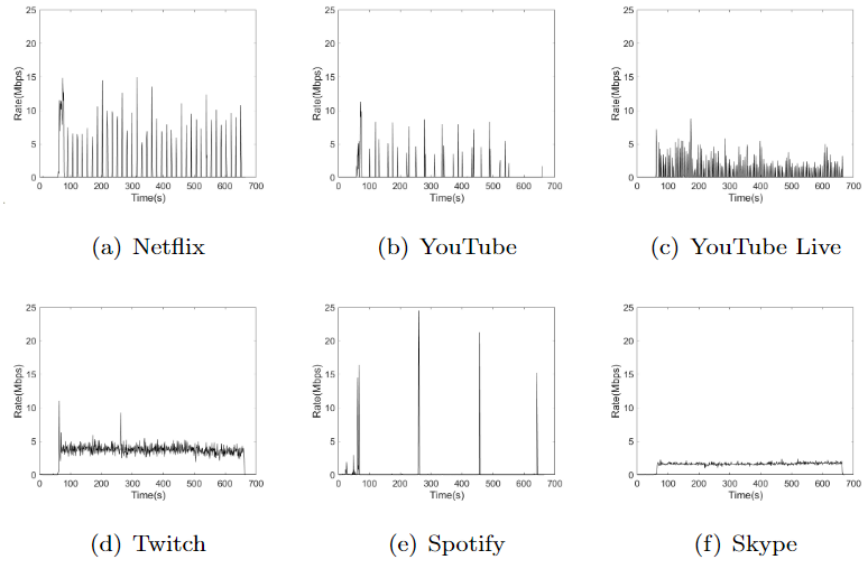

Figure 3.2 Instantaneous traffic rate examples for various applications

deviation of inter-arrival time, which are summarized in Table 3.1. Among these features, total size greatly depends on the size of the video content and it cannot be reliably used to distinguish between different traffic categories, let alone individual applications. The DL/UL ratio is perhaps the only useful feature to distinguish TC traffic from others. However, this ratio requires monitoring and handling of flows in two directions and creates a dependency on these bi-directional flows, consequently, increasing the implementation complexity of a classifier using this feature. Packet-level features also have limited usefulness, since they also depend on the transport layer mechanisms such as flow control and congestion control along the ETE path. The mean traffic rate could be a promising feature which addresses the overall behaviour of the traffic source while considering all parts of the ETE path and including the effect of different video and audio codecs; however, according to our traffic data, usefulness of mean rate also seems marginal.

Based on the above observations, we next study the instantaneous traffic rates of the applications as a function of time. For this purpose, we consider only the DL traffic rates, since all multimedia traffic has a DL component whereas only some of them (e.g., TC category) have a considerable UL traffic. As seen in Fig. 3.2, different applications exhibit different DL traffic rate patterns: The two VOD applications have periodic burst-silence cycles with different frequencies and amplitudes. Skype and Twitch share a pattern of oscillation over a baseline with different amplitudes. YouTube Live traffic, albeit being a VLS application akin to Twitch, shows more of a VOD-like behavior with a much higher frequency of bursts. As for the non-video traffic: Spotify exhibits occasional instantaneous peaks when the music player

Table 3.1 Feature analysis based on 300 minutes of traffic flow per application

Application	**Flow Level Features**					**Packet Level Features**		
	Direction	Total Size (Mbit)	Rate μ (Mbps)	RTZ	DL/UL	Length σ (kb)	Interarrival μ (s)	Interarrival σ (s)
Netflix	DL	647	0.822	406	20.67	1.903	0.0144	0.395
	UL	31.3	0.0388	387		0.084	0.0416	0.551
YouTube	DL	635	0.880	193	41.50	4	0.0132	0.337
	UL	15.3	0.0212	351		0.093	0.0761	0.772
YouTube Live	DL	1450	1.86	193	36.80	4.037	0.0061	0.241
	UL	39.4	0.0486	203		0.1	0.0356	0.558
Twitch	DL	1080	1.56	0	14.08	1,054	0.0068	0.059
	UL	76.7	0.107	0		0.077	0.0151	0.709
Spotify	DL	144	0.205	469	16.60	1.749	0.514	0.788
	UL	8.67	0.0123	347		0.081	0.1193	0.911
WhatsApp	DL	84.23	0.1404	300	N/A	4.421	0.1974	1.357
Skype	DL	1580	2.19	0	1.59	0.996	0.0037	0.251
	UL	993	1.38	0		0.618	0.0049	0.091

is about to switch to the next song and therefore downloads it in a single burst; whereas WhatsApp has a similar pattern representing the occasional download of data and messages.

We also consider the applications' instantaneous traffic rate curves as time-domain signals and check various representative time-domain features over these signals such as burst size, inter-burst arrival and return to zero (RTZ). Note that we omit frequency-domain features of these signals on purpose, since any such feature would require signals of longer time periods which would seriously reduce the timeliness of any traffic classifier built on top of them. In this context, we define the burst size by integrating instantaneous traffic rate signals over short time intervals, i.e., over burst durations such as 0.02 or 1 second. Inter-burst time is defined as the period between two consecutive bursts. Although the burst size-based analysis is fruitful to differentiate VOD and S&MS traffic, the base component of the VLS and TC traffic flows limits its use for these categories of traffic. Regarding the inter-burst time, selection of burst size in calculating this metric appears to be quite challenging. To remedy the shortcomings of the burst size-based analysis, we consider the RTZ pattern of each signal that is defined as the number of times the signal returns to zero traffic rate. As shown in Table 3.1, this feature clearly differentiates between Twitch and Skype traffic. However, RTZ does not seem to be sufficient on its own in differentiating all these different multimedia application categories, let alone identifying applications.

In summary, patterns of instantaneous traffic rates as signals in time domain visually demonstrate significant discrimination power in terms of traffic classification among the considered applications, cf. Fig. 3.2. However, it is still not possible to identify a set of features which classify all of the applications well. Although several features (cf. Table 3.1) are intuitive and discriminative regarding certain applications, they do not fully exploit the information present in the rate, because they are essentially hand-crafted and means of dimensionality reduction to provide computational and storage efficiency. For this reason, instead of feature extraction, we treat the rate

signal as a stochastic process and model it as a first order Markov with quantized amplitudes in discrete-time. The generalization to higher order and finer quantization is straightforward. To this end, we provide the problem description in the following and then explain our Markov modeling of traffic rates.

3.2 Problem Description

In the scope of this thesis; with the proposed dataset, seven class classification problem is handled via supervised learning algorithms based on DTMC models of traffic flows. To detect classes Netflix=1, YouTube=2, YouTube Live=3, Twitch=4, Spotify=5, WhatsApp=6, Skype=7, the signal a_n, that includes one or more traffic flow features is observed during period of L seconds. By looking at the traffic patterns in training set, general behaviours of applications are caught. Then the traffic flows with length of L that is kept in the test set, is predicted by the proposed algorithms after training process.

We point out that the streaming application might well be non-stationary and switch from one type, i.e., class, to another during the time course of observations. For this reason, we opt to process the data a_n by a sliding window (with the length $L/2$ in time) and keep only $L \times f_s$ samples per window for simplicity. Here, f_s is the sampling frequency in Hz, and the sampling should be understood together with integration: the sampled value at a time is the result of the aggregation of the sizes of all of the packets received after the last sample. We gain two benefits as the window length L gets smaller: i) the windowed stream can be better assumed to be from (or dominated by) a single application and ii) the timeliness of our classification scheme improves -however- at the cost of perhaps degrading classification accuracy (since the amount of observations also decreases) especially beyond a certain point. In this study, although we allow switching among applications, we assume that the user does not stream from two or more applications at the same time; or if she/he does, then we assume that one of the streaming applications dominates the traffic. We regard the generalization to the simultaneous streaming case as future work.

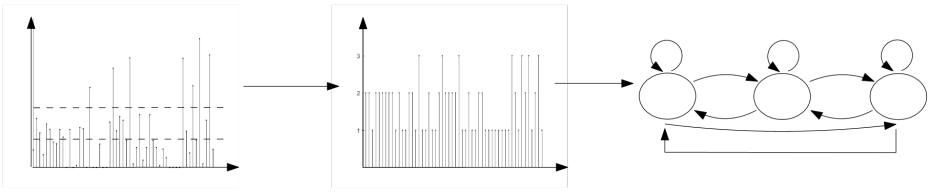

Figure 3.3 Min-max quantization

3.3 Markov Based Traffic Classifiers

In this section, we present two novel markov classifiers, where packet-based features are converted to Markov chains to utilize sequentiality. Applying the same approach, the two classifiers use different number of traffic features as their input.

3.3.1 k Nearest Markov Components (kNMC)

Multimedia traffic datasets considered in this thesis are collected from WiFi networks via packet sniffing/capturing tools, such as Wireshark, resulting in packet-based traffic data. The preprocessing approach described here can be applied to any packet-based dataset.

In DTMC model, first the packet-based traffic is converted to flow-based traffic by aggregating the total number of bits of downlink packets per a fixed flow interval, T, which is equal to $1/f_s$, f_s being the sampling frequency. At the end of the conversion, an L seconds-long packet-based downlink data becomes a flow-based data. This is a one-dimensional (1-D) discrete signal of rates, a_n, whose length is equal to $L \times f_s$. For example, if the duration, L, of the packet-based traffic is 600 seconds and the applied sampling frequency, f_s, is 10 Hz; then, the corresponding flow interval, T, is equal to 0.1 seconds and the length of a_n is $600 \times 10 = 6000$.

After these two stages, the final dataset is obtained as follows : $\{x_{n,k}, y_k\}_{k=1}^{N_{sum}}$, where $x_{n,k}$ is the k^{th} window/instance of the dataset with $l_w \times f_s$ samples, y_k is the corresponding application label for the k^{th} instance, and $N_{sum} = \sum_{i=1}^{N_a} N_w(i)$. Here, x_n is a list of N_{sum} elements and each element of the list is a $1-D$ list (list of scalars) of length $l_w \times f_s$. In addition, $x_{n,k}(a)$ refers to a^{th} sample of the k^{th} instance of x_n and $x_{n,k}(a) \in \{1, 2, \cdots, B_T\}$, where B_T is the upper bound for the number of

bits that can be downloaded in T seconds.

Once the preprocessed version of the dataset is obtained, each instance of x_n is modeled as a first-order DTMC, whose main parameter is the number of states, N_s. Each sample of an instance is mapped to one of N_s state groups. When the inputs of the mapping are one dimensional signals (e.g. rate signals), partitioning the input space turns into creating amplitude intervals for each state group. The partitioning method applied in [42] is k-means clustering algorithm.

Once the intervals are set, the instances of x_n can be quantized into digital signals (i.e. state sequences) and this quantized version of x_n is called q_n. $\{x_{n,k}, y_k\}_{k=1}^{N_{sum}}$ becomes $\{q_{n,k}, y_k\}_{k=1}^{N_{sum}}$ where $q_{n,k}$ is the k^{th} instance of the dataset. In addition, $q_{n,k}(a)$ refers to a^{th} sample of the k^{th} instance of q_n where $q_{n,k}(a) \in \{1, 2, \cdots, N_s\}$.

As the final stage of DTMC modelling, an $N_s \times N_s$ state transition matrix, M is formed for the each instance of q_n. The entries of the matrix, M_k for the k^{th} instance can be obtained as:

$$\begin{aligned} M_k[i,j] &= P_{q_{n,k}(a+1)|q_{n,k}(a)}(j|i), \\ \forall (i,j) : ((i,j) &\in \{1,2,\cdots,N_s\} \times \{1,2,\cdots,N_s\}), \end{aligned} \tag{3.1}$$

where $P_{q_{n,k}(a+1)|q_{n,k}(a)}(j|i))$ is the probability of observing state-j at $q_{n,k}(a+1)$, given that $q_{n,k}(a) = i$. Therefore, the dataset at the end of DTMC modeling is obtained as, $\{M_k, y_k\}_{k=1}^{N_{sum}}$.

Based on DTMC model, in [42] three classifiers are proposed, out of which, kNMC is the best-performing one. Since kNMC exploits the idea of nearest neighbors, first a distance definition is required between a train instance $x_{n,k}$, and a test instance $x_{n,i}$. kNMC uses a likelihood-based distance as defined below:

$$d(x_{n,i}, x_{n,k}) \triangleq -\log(P_{X_n}(q_{n,i}, M_k)), \tag{3.2}$$

where $P_{X_n}(q_{n,i}, M_k)$ refers to the likelihood of M_k being generated from $q_{n,i}$. Since M_k is the state transition matrix of $x_{n,k}$ and $q_{n,i}$ is the state sequence generated from $x_{n,i}$, likelihood can be computed as:

$$P_{X_n}(q_{n,i}, M_k) \triangleq \prod_{a=1}^{l_w f_s - 1} M_k[q_{n,i}(a), q_{n,i}(a+1)]. \tag{3.3}$$

As stated in (3.2), the distance is defined as the negative log of the likelihood in (3.3). The log operation prevents computational underflows and the negative sign

helps to keep the notion of the distance. For example, as the likelihood between two instances increases, the distance between them should decrease.

Having defined distance, building a classifier based on this distance is straightforward: For each test instance, k-nearest train instances form a neighborhood set based on (3.2). The decision is made by the equally-weighted votes of those k train instances, since in [42] it is assumed that difference in distance within a neighborhood is not significant enough to affect the final output of the classifier.

3.3.2 Three Dimensional k Nearest Markov Components (kNMC-3D)

As described in the previous section, kNMC uses only one feature of the downlink traffic, which is the number of bits in a fixed interval, i.e. the bit rate. Since the dimensionality of the feature space is one, it can also be called as kNMC-1D. By including more features, one may increase the dimensionality to n to obtain kNMC-nD. In this section, we introduce kNMC-3D by including two more features, inter-arrival time and packet rate, in addition to the bit rate feature.

In this work, we propose to increase dimensionality, as more features might provide more sensible information to identify the traffic pattern. However, this increase should be done carefully without facing the curse of dimensionality. Therefore, before explaining the reasoning behind the selection, we list and define the possible candidates for flow-based features as follows:

- **Bit rate (bps):** Total number of bits of downlink packets per T seconds-long flow interval.
- **Inter-arrival time (s):** Average inter-arrival time between the downlink packets in the flow interval.
- **Packet rate (pps):** Total number of downlink packets in the flow interval.
- **Downlink/uplink ratio:** Ratio of the lengths of downlink and uplink packets in the flow interval.
- **Average packet length (bits):** Average length of the packets in the flow interval.
- **Variance in packet length (bits):** Variance in the length of the packets in the flow interval.

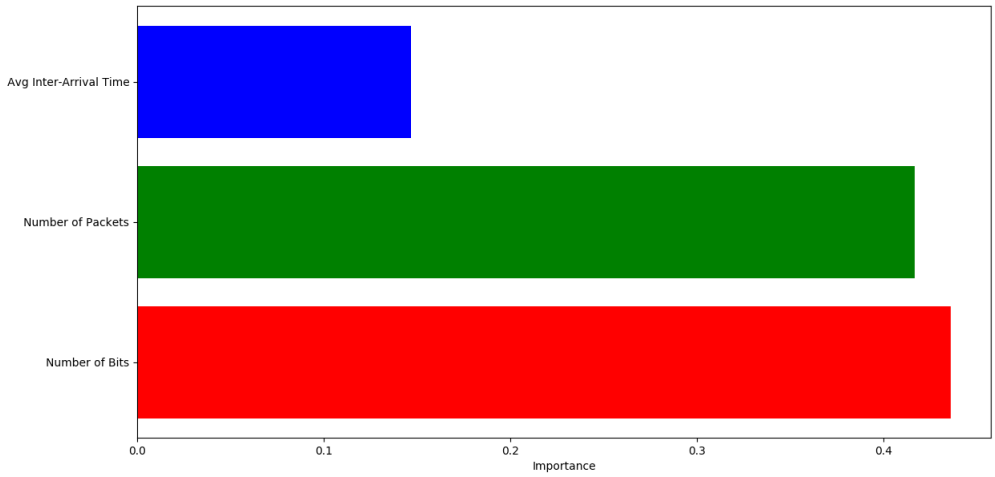

Figure 3.4 Feature Importances in kNMC-3D Model

Since it would not be desirable to design our method as sensitive to the uplink traffic and the individual packets of the flow, we eliminate the features of downlink/uplink ratio and variance in packet length. On the other hand, the first three features already cover the information from the remaining ones. For example, the average packet rate can be calculated as the bit rate divided by the packet rate. Therefore, in our method design, we choose to use the bit rate, inter-arrival time and packet rate features. Although we use only these features in the presented study, one can straightforwardly incorporate any other features into our method kNMC-3D that we next explain below. After we decide on the feature set, importance of three selected traffic features are calculated to find out each traffic feature's contributions. The term "feature importance" refers measuring the contribution of each input feature to the final performance for a given model. A higher score indicates that the particular characteristic will have more of an impact on the model being used to predict a particular variable. In kNMC-3D model, calculating feature importances for target classes are not straightforward since kNMC-3D is not an ordinary learning model. For simplicity, all combinations of the feature set are tried rather than applying well-known feature importance calculation techniques such as linear regression feature importance [43] [44] [45], random forest feature importance [46]. By looking at the average test accuracies of all scenarios the final feature's importances are obtained. The main methodology relies on changing values of features one by one randomly and investigating the changes. The applied methods are similar to permutation feature importance calculation[47]. The importance of the selected features in kNMC-3D model can be observed in Figure 3.4. According to the results, number of bits and number of packets traffic features have nearly 40% importance in kNMC-3D model. Average inter-arrival time does not have contributions as much as the other two features but it is not negligible. Therefore we decided to continue with all of three

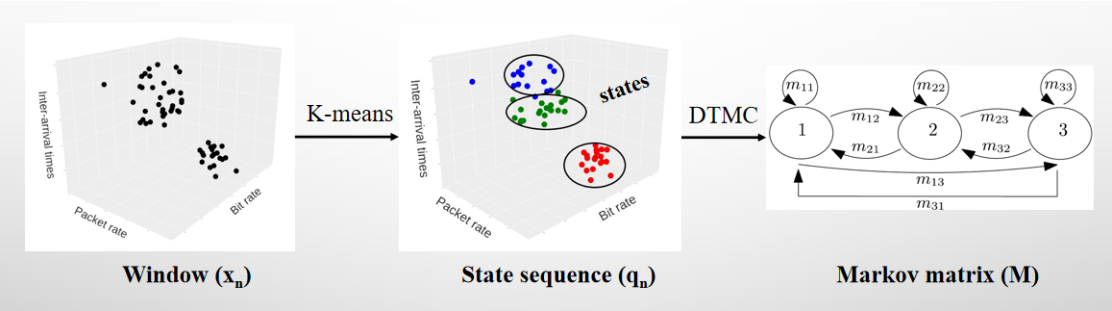

Figure 3.5 3D clustering via k-means

features.

Modeling the rate signals as a DTMC has been introduced in Section 3.3 for the case of using one feature (kNMC-1D). In this section, we explain the details of generalization to the case of multiple features, leading to our method kNMC-3D.

Since the multidimensional approach expands the dataset, we re-define it as: $\{x_{n,k}, y_k\}_{k=1}^{N_{sum}}$, where x_n is a list of length N_{sum}, and each instance of the list consists of 3D vectors of length $l_w \times f_s$. In order to model each instance of x_n as a DTMC, x_n should be mapped to one of N_s state groups. Hence, for the case of multiple features here, inputs of the mapping are 3D signals (e.g. {bit rate, inter-arrival time, packet rate}) and the outputs shall be a partitioning of the input space, i.e., a vector quantization, for which the k-means clustering can be used. Before the clustering operation, and unlike kNMC-1D, min-max normalization is used for the chosen features to bring them to the common scale of $[0, 1]$.

there should be multidimensional clustering and each representative value of three features(3D) of the same time instance should be common scale. Our features (i.e. number of bits , number of packets and inter-arrival times) are numeric features and their ranges are different from each other. Therefore, normalization is needed to obtain more reliable clusters and to prevent distortions. In the scope of this paper, min-max normalization and mean normalization are tried and min-max normalization that scales the all values between 0 and 1 is decided to be used. Each feature set is normalized with min-max normalization in themselves.

Normalization is followed by the k-means clustering and quantization steps to obtain a DTMC based on multiple features. We have two main steps in the k-mean clustering. First, each vector instance is assigned to the closest cluster center (which are randomly initialized). Second, the cluster centers are re-calculated based on the assigned instances. This process of two steps is repeated in an iterative manner at the convergence of which clusters in the data are found. The output is a sequence of cluster indices (which provides the state sequence in our DTMC) corresponding

to the sequence x_n of the 3D networks features. The rest of the kNMC-3D steps are same as kNMC since the multidimensional sequence x_n is reduced to 1D q_n. In the next step, the state transition probabilities and likelihood distances are calculated as described in Section 3.3.1. Then, the final classification follows with the k-nearest neighbours approach as in kNMC.

4. ECOC BASED kNMC-3D CLASSIFIERS

Error-correcting output codes (ECOC) is a technique that solves multiclass problems with multiple binary classifiers. By this way, the errors of the classifiers can be fixed with other binary classifiers. The usage of binary classifiers also prevents the negative effects of class variances in multiclass problems. There are two main motivations for using ECOC. The first one is increasing multiclass classification accuracy. The second one is utilizing classifiers such as SVM and logistic regressions.

In this chapter, the ECOC framework is applied to a given problem by using previously defined classifiers. Since ECOC-based solutions are able to increase the performance of multiclass classifiers, they are popular in different areas. [48] which introduces ECOC first time, applies ECOC on both neural networks, decision trees and show that ECOC has a positive effect on accuracies in multiclass problems. The authors also explain the main design rules for specific ECOC matrices and three code-constructing strategies are introduced: BCH, Exhaustive Codes, and Randomized Hill Climbing. [49] presents a novel ECOC strategy that segments the base subproblem (SECOC) using subclass partitioning.

The ECOC design can be done by two different ways: problem-dependent design and problem-independent design. The problem-independent designed ECOCs are general solutions that work with fixed code matrices. Commonly used problem-independent ECOC methods are: One versus One (OVO), One versus All (OVA), Dense Random and Sparse Random introduced respectively by [50], [51], [52], [53]. These models can be applied to any problem with the arrangement of the code matrix regarding the number of classes. In [54], ECOC is applied to document categorization. Face identification work based on ECOC can be seen in [55]. [56] offers various ECOC-SVMs designs that have different code lengths for solving remote sensing image recognition problems. Recommended classifiers that work on different datasets are compared in terms of prediction accuracy versus code length. [57] tries to solves traffic classification problems with neural network ensemble and error-correcting output codes. It works on five classes: BT, HTTP, SMTP, HTTPS, and others. The performance of their ECOC-NNE is 92% and it outperforms three

different benchmark methods. Confusion matrix-based ECOC design for pattern recognition is presented in [58]. The authors consider class similarities in the matrix design process. In the beginning, easily separated classes are selected and ECOC matrix is updated adaptively by following Fisher principal.

In the ECOC approach, the design of the ECOC code matrix is the most important point. Each class has a codeword which is the row of the ECOC matrix. The columns of the ECOC matrix correspond to sub-classifiers that are also called as dichotomizers. For each sub-classifier, relabelling operation is done and the base classifier is converted into binary sub-classifiers. The construction of ECOC matrix and codewords is called coding operation. After the learning process is completed, a bit vector that has a length of codewords is predicted. After predicted bits are compared with representative codewords of classes, the closest class is selected in the test phase. This part is called the decoding stage of ECOC-based learning. In this thesis, the various applications of ECOC on the kNMC-3D classifier which was introduced in Chapter 3 are considered for further improvement.

Error-correcting output codes is a framework that gives the opportunity to fix the classifier's errors with other classifiers. While dealing with multiclass problems, ECOC offers to use multiple binary classifiers. As explained previously, the main learner of our ECOC designs will be kNMC-3D. In the encoding process of ECOC, the labels of kNMC-3D are changed to turn to binary. Since kNMC exploits k nearest neighbors, relabelling operation has a great impact on the results. The most important point for kNMC-3D-based ECOC design is changing of clusters. Although kNMC-3D uses unsupervised k-means which is independent of labels, the construction of clusters becomes completely different where data of some classes are annotated with label 0. It means that the data of 0 classes is out of the training process.

kNMC-3D with Soft Decision (kNMC-3D-SD)

The standard kNMC-3D directly predicts the class label in the multiclass case as defined. In binary case, kNMC-3D makes a prediction as a negative or positive class. For simplicity, we call this standard binary kNMC-3D as kNMC-3D with hard decision (kNMC-3D-HD) since it gives direct results. Sometimes the prediction sequence of sub-classifiers Y_n can have the same distance to codewords of two different classes although their codewords are well separated. To not face that type of problems, kNMC-3D with soft decision (kNMC-3D-SD) binary classifier is also introduced. This classifier is a slightly different version of kNMC-3D-HD. In the decision stage of kNMC-3D, the one that works with soft decision saves the distances of k closest train instances of positive and negative classes rather than saving classes of the k

closest train entries. As described in Chapter 3, kNMC-3D makes a prediction by looking at dominating class in the k closest classes. However, kNMC-3D-SD calculates the average closest distances of positive and negative classes. If we assume that m_1 is the average distance of a given test entry for the positive class and m_2 is the average distance for the negative class, the prediction p for kNMC-3D-SD will be

$$p = 2m_2/(m_1 + m_2) - 1 \tag{4.1}$$

By this way we end up with a score between 1 and -1 with kNMC-3D-SD. Despite the change in the decision state of kNMC-3D, the main approach will be the same because the clustering and DTMC modeling is not affected. With kNMC-3D-SD, more sensible results could be produced thanks to numerical results. Even though kNMC-3D-Soft solves complex cases such as the one defined before, it could be misleading in some other cases. In multiclass case, it is observed that using kNMC-3D with hard decision mechanism performs better. Since kNMC and kNMC-3D use a local approach (looking at the classes of k training instances); that situation could be considered as expected. However, using kNMC-3D-SD with ECOC schemes may perform better due to the nature of the ECOC framework. Therefore trying both kNMC-3D-SD/HD as base learners and comparing the results might be a more sound approach.

4.1 Problem-Independent ECOC Schemes

There are problem-independent ECOC schemes such as OVO, OVA, Sparse Random, and Dense Random. These approaches present fixed ECOC designs for every problem. Because OVO and OVA are most popular, in the application phase for a given problem, the two of them are preferred. Their coding strategies and the number of sub-classifiers of the two methods are different. In the scope of the thesis, two different applications for OVO and one for OVA schemes are used in the solution of the given traffic identification problems.

4.1.1 One Versus All (OVA)

OVA classification approach combines the information of $n = 7$ (number of classes) sub-classifiers. Each sub-classifier separates a class from the other six classes. Assume that sub-classifier c_i is a separator for the i^{th} class, then in the corresponding ECOC matrix, i^{th} class is labeled as 1 and other rows labelled as -1. Because the ECOC matrix includes 1's and -1's only, that type of encoding is called as binary encoding. The sub-classifiers of OVA have nearly the same training section as the base multiclass kNMC-3D classifier. Since state quantization levels of kNMC-3D are decided with unsupervised k-means and all the data are included in this procedure, the states will be the same in all sub-classifiers training. The main difference between sub-classifiers and base multiclass learner will happen in the last decision stage of the algorithm due to relabelling. The OVA application provides the final prediction in the test phase with the following equation.

$$Y_{end} = \underset{i \in \mathcal{I}}{\operatorname{argmax}}\, c_i(x) \tag{4.2}$$

where $I = 1,2,3....n$, Y_{end} is final prediction and x is the input test instance.

4.1.2 One Versus One (OVO)

OVO approach uses $nx(n-1)/2$ sub-classifiers. The duty of each sub-classifier is to provide a separation between two classes. In the OVO approach unlike OVA, ternary coding method is applied. Ternary coding uses 1,0,-1 in the encoding process. Each sub-classifier considers just two classes and ignores the data of other classes that are encoded with 0 in the binary classification. In the designing of the OVO-ECOC matrix for each column c_i positive and negative classes are labeled as 1 and -1, all of the other classes are labeled as 0. Labeling with zero means that the sub-classifiers have no info about 0 classes and do not process the data of that classes. In our application of OVO within $N_c = 21$ sub-classifiers, OVO covers all the pairs. In OVO case despite we have a single ECOC matrix, the two different versions of kNMC-3D are employed as a base learner. Therefore two different OVO version is introduced: One Versus One with Hard Decision (OVO-HD) and One Versus One with Soft Decision (OVO-SD). OVO-HD work with kNMC-3D-HD in its sub-classifiers. If we assume that $Y_n \in \{1, -1\}$ is a predicted bit sequence that has a length of N_c, for a single instance and Cw_i is the i^{th} class codeword which refers to

i^{th} row of the ECOC matrix, the distances between prediction and each class are calculated as follows:

$$d = \sum_{k=1}^{N_c} -Y_n[k]Cw_i[k] \tag{4.3}$$

For $i = 1,;n$ distances are calculated and i^{th} class with the minimum distance becomes the final prediction. The distance calculator is based on Loss Based Decoding. The second version of OVO which is OVO-SD; uses the same ECOC matrix with OVO-HD. As expected, the biggest difference of the OVO-SD is the usage of kNMC-3D-SD as a base learner. Therefore the Y_n now includes values between 1 and -1, it is not a bit sequence anymore. In the decoding process of OVO-SD, Attenuated Euclidean decoding (AED) [59] is preferred. The distance between codewords and Y_n is calculated with.

$$d = \sum_{k=1}^{N_c} Y_n[k] - Cw_i[k](Y_n[k] - Cw_i[k])^2 \tag{4.4}$$

where Cw_i is i^{th} codeword and k is bit position. After distances are calculated, the class of the codeword that has the smallest distance is predicted.

4.2 Problem-Dependent Confusion Based ECOC Schemes

OVO and OVA classification schemes use fixed ECOC matrix designs and decoding rules. In this section, the special ECOC matrix designs that are custom constructed for the proposed problem will be given. Two different approaches will be presented in the scope of problem-specific ECOC schemes. These are Confusion-Based ECOC Schemes and Two-Level ECOC Schemes.

Table 4.1 Confusion at the application level (Classifier: kNMC-3D)

Act \ Pred	Netflix	Youtube	YT Live	Twitch	Spotify	WhatsApp	Skype
Netflix	95.17	2.76	0.69	1.03	0.34	0.00	0.00
Youtube	0.34	90.69	6.90	0.00	2.07	0.00	0.00
YT Live	1.38	0.00	96.55	0.34	0.00	0.00	1.72
Twitch	3.45	0.00	0.00	94.48	0.00	0.00	2.07
Spotify	1.03	18.62	3.45	0.00	67.24	9.66	0.00
WhatsApp	3.10	10.34	4.83	0.00	12.76	68.97	0.00
Skype	2.41	0.00	10.34	0.34	0.00	0.00	86

YT Live: Youtube Live Act: Actual, Pred: Prediction

4.2.1 Confusion Based ECOC Schemes (CB-ECOC)

OVO and OVA are well-performed methods for simple multiclass classification problems. However; for complicated problems, custom-designed ECOCs according to the problem requirements could end up with more successful results. Or custom design ECOC could be more efficient due to the usage of the low number of classifiers. Also custom designed ECOC matrix is flexible, unlike OVO and OVR cases. Sometimes the problem does not require all pairwise sub-classifiers because the complexity gets higher. Hereby custom design ECOCs are preferable solutions. While designing an ECOC matrix, there are three important rules to be considered. i) The rows of the matrix should be well separated. ii) The columns of the code matrix will be well separated and uncorrelated. iii) The error of the binary classifiers should be low. To satisfy the third rule we used our own classifier that provide very successful results in terms of application and category accuracies. Therefore this rule is already fulfilled. The first rule is the most critical point for an ECOC matrix. If the two rows of the matrix which are codewords of classes are so close to each other; the prediction probabilities of referred classes could not be high. The ECOC frameworks are able to fix $d/2$ errors where d is equal to Hamming distance between the two closest codewords. Sometimes to ensure row separation, unnecessary classifiers can be added to the model. The second rule is almost as important as the first rule for ECOC design patterns. Adding same columns or correlated columns to the matrix seems as unimportant situation since they carry the same information, it can lead to devastating performance effects on results. As the columns that are sub-classifiers could fix errors, they might cause errors too. Therefore adding the same column to the matrix can strengthen errors. As a result column separation is also very important for matrix design.

Table 4.2 Matrix of Generic Sub-Classifiers (M_g)

Class	c_1	c_2	c_3	c_4	c_5	c_6	c_7
Netflix	1	-1	-1	0	-1	-1	-1
Youtube	0	1	−1	−1	0	−1	−1
Youtube Live	−1	0	1	−1	−1	−1	0
Twitch	−1	−1	−1	1	−1	−1	−1
Spotify	−1	−1	−1	−1	1	0	−1
Whatsapp	−1	−1	−1	−1	−1	1	−1
Skype	−1	−1	0	−1	−1	−1	1

A confusion matrix is a useful success indicator for classification schemes. It shows classification errors in detail. For a confusion matrix F, $F_{i,j}$ show the prediction ratio of j^{th} class where the real class is i^{th} class. The elements of the confusion matrix also represent the correlation (here the word correlation is used as similarity, where high correlation/similarity between two classes leads to high confusion in the classification) between classes. There are similarities between classes where classification errors occur mostly. Therefore the construction of ECOC by considering the confusion matrix could be a smart solution to fix errors. Our ECOC approach mainly focuses on this idea. Confusion-based ECOC design is done in two stages. In the first stage of design methodology, n sub-classifiers called as generic sub-classifiers are constructed. The duty of generic sub-classifiers is to separate each class from other classes except the class that has the highest correlation. For each class, there is only one generic sub-classifier. If we assume that c_i is a generic sub-classifier for i^{th} class and j^{th} class is the class that has the highest cross-correlation with i^{th} class; the i^{th} row of c_i is labeled with 1, the j^{th} row is labeled with 0 and other rows are labeled with -1. By this way; the generic sub-classifier of each class ignores the data of the class that cause trouble at discrimination of the targeted class. The M_g matrix which is the end product of the first stage, includes all generic sub-classifiers. The construction rule of M_g matrix can be seen in Algorithm 1.

The first phase of the ECOC matrix construction is done for reaching a general distinction between classes that share a low correlation with each other. However, in the specific cases where cross-class similarities are too high, the generic sub-classifiers have a high possibility of error. For example, when we proceed with the c_1 of the M_g matrix; if the test entry is from the Youtube class, its predicted bit for c_1 would be one due to its high correlation with Netflix. Even the corresponding bit is not important because this position of the Netflix codeword is 0. The mentioned prediction of bit causes the prediction sequence Y_n to become closer to the Netflix codeword. Therefore to fix these types of mistakes the groups of sub-classifiers called specific sub-classifiers are prepared. Again in the preparation operation, the cross-class correlation information that can be obtained from confusion matrix F

Algorithm 1 Construction of generic sub-classifier matrix (M_g)

```
for i ∈ {1,...,n} do
  for k ∈ {1,...,n} do
    if F[k][i] is highest in column then
      M_g[k][i] ← 0
    else
      if i = k then
        M_g[k][i] ← 1
      else
        M_g[k][i] ← −1
      end if
    end if
  end for
end for
```

4.1 is used. In the case of specific sub-classifiers, the encoding approach that is exactly opposite of the design way of M_g is considered. Dichotomizers which are pairwise (1vs1) sub-classifiers for highly correlated classes are added to the specific sub-classifiers group. If $F_{i,j}$ is bigger than 5% where i is not equal to j, the pairwise sub-classifiers of i^{th} and j^{th} classes for $i, j \in 1, 2....n$ are included in the specific sub-classifier groups. While we handle the easier classification cases with the generic sub-classifier group which considers the data of almost all classes except one, for the complex cases pairwise classifiers seem to be better option. This is because the pairwise classifiers have a chance to create clusters that are more sensitive to considered pair classes. Hence the desired distinction can occur between classes with higher expectations. The specific sub-classifiers are included in matrix M_s and are listed in Table 4.3. For the final ECOC matrix, the combination of M_g and M_s is preferred.

Like OVO case, two different CB-ECOC classifiers are presented: CB-ECOC-HD and CB-ECOC-SD. The CB-ECOC-HD classifier employs kNMC-3D-HD as the base learner. As the ECOC matrix, it uses CB-ECOC. Like OVO-Hard, loss based decoding strategy with a linear loss function is used in the decoding process. The distance calculation formula can be seen in Equation 4.3. Unlike CB-ECOC-HD, the CB-ECOC-SD classifier employs kNMC-3D-SD as the base learner. As the ECOC matrix, it uses the same matrix with CB-ECOC-HD. Like OVO-SD, Attenuated Euclidean decoding strategy is used with a distance formula which is represented by Equation 4.4.

Table 4.3 Matrix of Specific Sub-Classifiers (M_s)

Class	c_1	c_2	c_3	c_4	c_5	c_6
Netflix	0	0	0	0	0	1
Youtube	1	−1	0	1	0	0
Youtube Live	−1	0	0	0	−1	0
Twitch	0	0	0	0	0	−1
Spotify	0	1	1	0	0	0
Whatsapp	0	0	−1	1	0	0
Skype	0	0	0	0	1	0

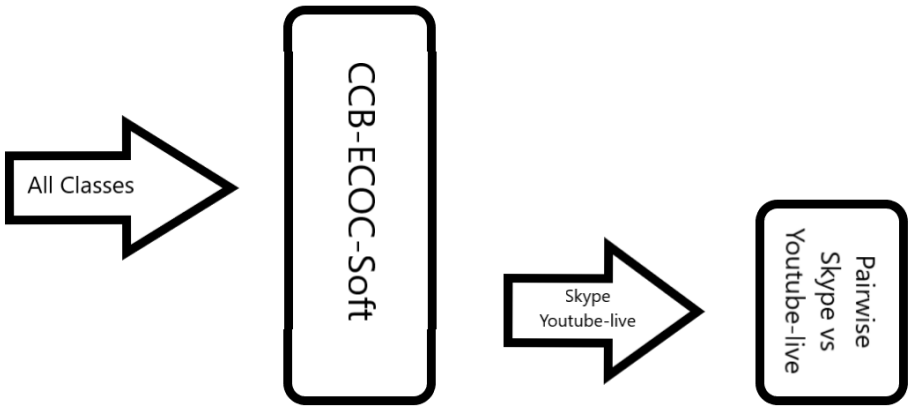

Figure 4.1 Two-level ECOC block diagram

4.2.2 Two-Level ECOC Scheme (2L-ECOC)

This classifier scheme is created by considering the classification results of the CB-ECOC-SD classifier. Again the confusion matrix of CB-ECOC-SD is deeply investigated to reach further solutions. When the results are observed, an interesting situation is seen. While CB-ECOC-SD fails for the identification of Skype, it achieves extremely high classification accuracies in other classes. More than half of the Skype test entries are predicted as Youtube-Live. However, we have a pairwise Skype vs Youtube-Live sub-classifier in our CB-ECOC matrix. The error could have been fixed within that sub-classifier. In the results of the CB-ECOC-HD classifier, the same situation did not happen. Since our distances are not integer in soft decision classifiers, other sub-classifiers might lead to a bigger numerical error. Despite the mentioned issue, in the identification of other classes performance of CB-ECOC-SD is excellent. The results can be seen in Table 5.11. For solving the Skype identification problem a few operations can be done. A sub-classifier that ensures Skype, Youtube-live distinction can be added to the CB-ECOC matrix but it could harm the identification of other classes and have bad effects on the solidity of the CB-ECOC matrix. Therefore an additional error-correcting classification level is utilized

in the classification process rather than changing the CB-ECOC matrix. By this way, two-level classification scheme is formed. The block diagram of the two-level ECOC classification scheme is represented in Figure 4.1. In the first level of classification, CB-ECOC-SD is directly applied. In the second level, a pairwise KNMC-3D classifier of Skype vs Youtube-live is applied on test entries that are predicted as Skype or Youtube-live in the first level with CB-ECOC-SD. Hereby most of the mispredictions are corrected.

5. EXPERIMENTAL RESULTS

In this chapter, the performance of the classifiers proposed in Chapter 3 and Chapter 4 are presented via detailed experiments. The accuracy of the classifiers is observed and evaluated by considering two different datasets. The first dataset is the main dataset of the thesis which is gathered by our team as introduced in Chapter 3. The second dataset is a dataset ISCXVPN [60] from the literature, which we use as a benchmark for further verification.

ISCXVPN dataset mainly includes packet captures of two types of Wi-Fi traffic, with VPN and without VPN. We focus on traffic without VPN since VPN encryption can affect the traffic patterns, which is out of the scope of our work. Therefore, without VPN eleven different applications are selected including Netflix, Spotify, Skype Video as in our dataset, plus new applications, Skype File, Facebook Chat, Hangouts Chat, Facebook Video, Hangouts Video, Facebook Audio, Hangouts Audio and Skype Audio. Since the ISCXVPN dataset has different types of applications that do not match the ones in our dataset, re-categorization had to be done. While VOD, S&MS and TC categories are also found in the ISCXVPN dataset. However, there is no sample for the VLS category in the ISCXVPN dataset; instead, Voice(VO) category which contains the interactive voice traffic is added.

5.1 Experimantal Setup and Paramater Optimization

The experimental setup for kNMC includes a preprocessing step applied to the raw dataset $\{u_{i,t}, l(u_{i,t})\}_{i=1}^{N_u}$ (where $u_{i,t}$ is a rate signal, $l(u_{i,t})$ represents the corresponding application label, $N_u = 30 \times 7 = 210$ and each $u_{i,t}$ is of $T = 600 = 60 \times 10$ seconds). The details of the raw dataset are explained in Chapter 3 and several examples are illustrated in Figure 3.2. As a result of the preprocessing consisting of sampling and window sliding as explained in Chapter 3 and partly illustrated in Figure 3.5, we ob-

tain the preprocessed dataset $\{(x_{i,n}, y_i)\}_{i=1}^{N_x}$ which is subject to classification (where $x_{i,n}$ is a windowed and sampled rate signal and y_i is the corresponding application label). For window sliding: the window length $W_l = 2 \times 60 = 120$ seconds and stride (amount of sliding) $W_s = 1 \times 60 = 60$ seconds, yielding $N_x = \left\lfloor \frac{T}{W_s} \times N_u \right\rfloor \simeq 2100$ (2093 to be more precise due to borders of sliding) instances in total with ~ 300 (299 to be more precise due to borders of sliding) instances per each application. Recall that such a windowing step with a sufficiently small window length is necessary to extract statistically stationary parts of the rate signal (such that the type, i.e., class, of the streaming application and also the corresponding subclass pattern do not change during the window, although an application can have various patterns), which is later followed by Lloyd-Max quantization for Markov modeling to yield $s_{i,n}$, as explained in Chapter 3 as well as Fig. 3.2. Note that an instance $x_{i,n}$ is of length $W_l f_s = 120 \times f_s$, where f_s is the sampling frequency in Hz. For kMMC-3D, the same preprocessing operations are done. The biggest difference is that the $\{u_{i,t}$ is a 3D vector that includes bitrate, number of packets and inter-arrival times. In that case, sampling and windowing are applied to three different signals.

In order to assess the multiclass classification performance in our experiments statistically robustly, 10 random training (90%) and test (10%) splits of the preprocessed classification dataset $\{(x_{i,n}, y_i)\}_{i=1}^{N_x}$ are generated. To prevent information leak between the training and test sets, first the sessions (30 ten-minute long runs per each application) are split and only then the windows are extracted. This guarantees that windows of a session can be included in either the training or test, but not both. When there is a clear observation regarding the value of a parameter by manual inspection, the average (over 10 splits) accuracy in the test set is reported without cross validation; otherwise, a detailed 5-fold cross validation is conducted for assurance and then the average (over 10 splits) accuracy in the test set is reported. The results are provided both in terms of the overall average multiclass classification accuracy and the class specific average classification accuracy. The range of the reported accuracy is also provided with the min/max statistics in our validation/parameter optimization experiments or with standard deviation in our performance experiments. For parameter optimization, we report the overall average multiclass classification accuracy (out of 7 application classes and 10 splits); and for performance results later, we also report the confusion matrices as well as the class specific average classification accuracy in addition to the overall average multiclass classification accuracy.

We start with analyzing and optimizing the three important parameters in our experimental setup: the number N_s of the states in our discrete-time Markov chain (DTMC), the sampling frequency f_s and the number k of neighbors around a test

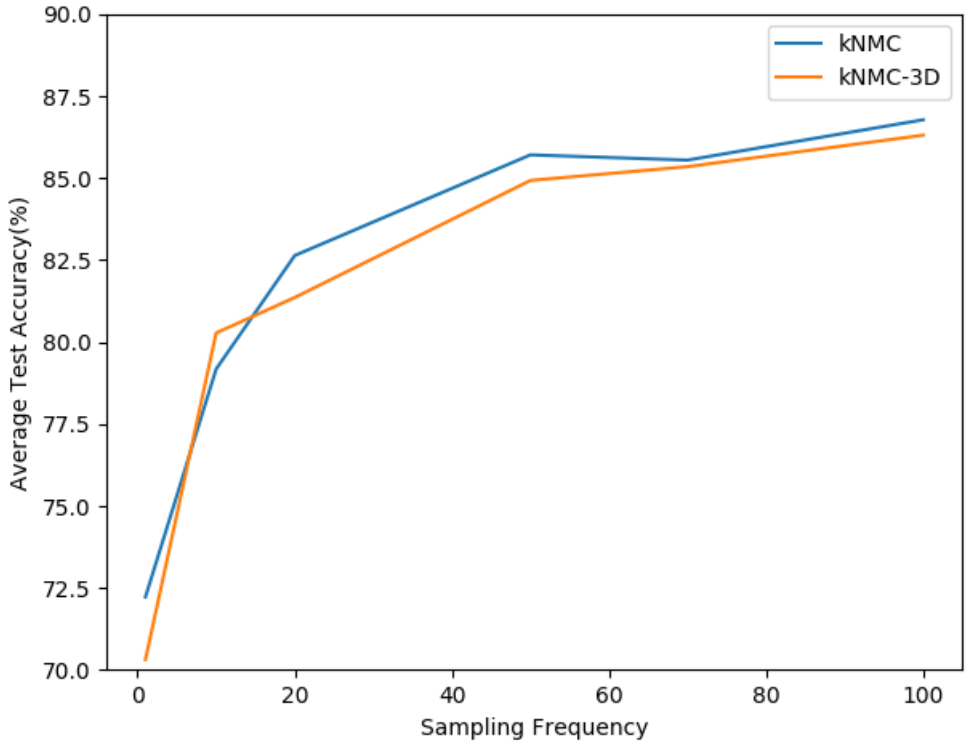

Figure 5.1 Average test accuracies for different sampling frequencies where $N_s = 8$ and $k = 3$, $N_s = 12$ and $k = 5$ in kNMC and kNMC-3D, respectively

instance x_n in the cases of the proposed classifiers kNMC and kNMC-3D. The effect of these parameters are quantified for optimization in terms of directly the end classification performance.

Figure 5.1 reports classification accuracies with kNMC, kNMC-3D for different sampling frequencies(f_s) where $N_s = 8 and k = 3$, $N_s = 12 and k = 5$ respectively. For both classifiers, increasing in sampling frequency provides an possitive effect on accuracies. This can be considered as an expected results because bigger sampling frequency refer to bigger resolution which give more info. On the other hand if the frequency become bigger, the complexity increase a lot. The increasing rate of the accuracy seem high until $f_s = 50$. Beyond that point there is a little improvement while f_s are moving to 100 from 50. In the results of both kNMC and kNMC-3D, same observation existed. Therefore choosing $f_s = 50$ is an efficient way to consider both performance and complexity. In the frequency optimization, the other two parameters N_s and k fixed.

The average test accuracies for different values of number of states(N_s) are shown in figure 5.2. For better state optimization fixed k and f_s are preferred. In both classifiers to the some number of states level, accuracies increase. For kNMC to the $N_s = 8$ accuracy increase. Adding more states do not improve the performance even

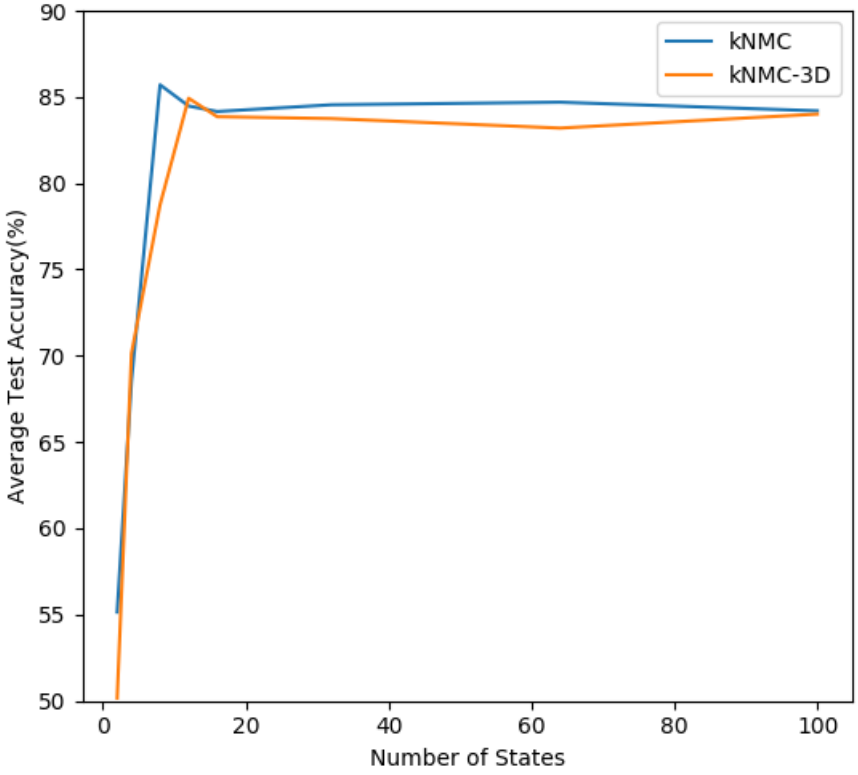

Figure 5.2 Average test accuracies for different number of states where $f_s = 50$ and $k = 3$, $f_s = 50$ and $k = 5$ in kNMC and kNMC-3D, respectively

Table 5.1 Overall average multiclass classification accuracy (%) for optimizing k, where $f_s = 50$ Hz for both classifiers, $N_s = 8$ for kNMC and $N_s = 12$ for kNMC-3D

k	**Application**		**Category**	
	kNMC	***kNMC-3D***	***kNMC***	***kNMC-3D***
1	84.74	84.67	88.58	90.56
3	84.99	84.93	89.1	91.13
5	85.71	84.64	89.11	90.67
7	83.45	82.80	86.90	89.24
9	81.87	81.75	87.03	87.15

it cause small decreasing effect. For kNMC-3D the same behavior are observed but max accuracy is achieved where $N_s = 12$. The kNMC-3D require more states for better performance because it works with information that combines three different features. To represent the effects of all features usage of more states in kNMC-3D could be considered as normal situation. Both classifiers provide bad performance near $50-60\%$ with low number of states. Therefore the optimization of the number of states is an important point for getting higher performances.

Another important parameter of the proposed DTMC based classifiers is k. It represent the number of neighbourhoods train instances in terms of markov components

that play a role in the prediction. Dominating class label in the between k neigbours train instances become the end prediction. Application and category accuracies of kNMC and kNMC-3D are reported in Table 5.1 for different odd k values, fixed f_s, N_s. kNMC perform bests where k is equal to 5 at the application level and where k is equal to 3 at the category level. Bigger k values than 5 provide $3-4\%$ worse performance. For $k=1$, performances are also not bad but using $k=1$ would not be good strategy due to decision based on only one train instances. For kNMC-3D, $k=3$ provide the most successful results at both category and application levels. Like seen in kNMC, bigger k values has an negative effects on accuracies. Eventually $k=5$ and $k=3$ are selected for kNMC and kNMC-3D. In the next results the results are presented with that k values.

5.2 Performance of DTMC Based Classifiers

In this section, we evaluate the performance of the proposed classification schemes in comparison to certain state-of-the-art classifiers: k-nearest neighbor classifier (kNN), support vector machines (SVM), multilayer perceptron (MLP), naive Bayes (NB), random forest (RF) and a decision tree (DT), cf. [24] for their uses for multimedia traffic classification on our dataset. These compared algorithms are all feature extraction-based classifiers, in which a feature vector is first computed from a window of traffic flow and then classified. The features that we use with these algorithms are described below in detail. As an alternative to traditional feature extraction, an autoencoder (AE) is also considered in the comparisons for learning the features in a data driven manner based on deep learning. Afterward, we present further experiments on the benchmark dataset from the literature, (ISCXVPN [60]).

We follow the same experimental procedure (regarding the performance evaluations as well as the parameter optimizations via cross validation) as the one that we describe at the beginning of the experiments section. All of our performance evaluations in this part are also presented in terms of the average multiclass classification accuracy along with its standard deviation (over 10 different training/test splits). For the new benchmark dataset, we increase the number of Markov states to $N_s=10$ (which is $N_s=8$ for our own dataset) and increase the neighborhood parameter k of kNMC and kNMP to $k=7$ (which is $k=5$ for our own dataset) as they are empirically observed to perform better. The parameters of the compared algorithms are all extensively 5-fold cross-validated for optimization. For example, in the case

Table 5.2 Overall average multiclass classification accuracy (%) at the application level in comparison to various state-of-the-art classifiers with Markov parameters as features

Classifier	$N_s = 4,\ f_s = 1$ Hz	$N_s = 8,\ f_s = 1$ Hz	$N_s = 4,\ f_s = 50$ Hz	$N_s = 8,\ f_s = 50$ Hz
kNMC	71.72 ± 4.01	75.17 ± 6.17	81.87 ± 3.93	85.71 ± 4.08
SVM	72.19 ± 2.48	74.33 ± 2.89	77.42 ± 5.87	79.50 ± 3.25
MLP	74.40 ± 4.12	76.34 ± 3.47	76.54 ± 5.36	78.91 ± 4.11
NB	61.12 ± 5.89	58.37 ± 7.44	63.55 ± 6.75	62.3 ± 3.45
RF	76.20 ± 5.52	78.55 ± 3.44	81.35 ± 3.37	84.50 ± 4.32
DT	73.45 ± 5.60	78.31 ± 3.57	77.9 ± 3.97	79.61 ± 6.67

of the introduced dataset, the cross validation typically returns: $k = 3$ for the number of the nearest neighbors that are checked in kNN, radial basis function kernel with the bandwidth $\gamma = 0.01$ and error cost $C = 200$ for SVM after normalization, 8 layers with ReLu activation and adam optimizer (adaptive step size $\eta = 0.005$ and epoch-by-epoch minibatch training with batch size 100) for MLP, a depth-10 decision tree with the gini index for splitting and 200 depth-6 trees for random forest with the gini index for splitting. As for the naive Bayes, we use Gaussian class densities with no priors. We find a similar setting for the new benchmark dataset as well, with the exceptions that $\gamma = 20$ and $C = 100$ for SVM with rbf kernel (without data normalization) and 100 depth-4 trees for random forest. These parameters are cross-validated again when the Markov parameters are used as features.

Table 5.2 compares our classification schemes with those state-of-the-art classifiers when the Markov parameters are used as features. For instance, for each window of a traffic flow, a Markov model is fit as described previously, the matrix of state transition probabilities is obtained and vectorized, and then used as a feature vector with SVM. We observe that, at the level of applications, the proposed kNMC outperforms not only kNMP (which is equivalent to kNN with the features of Markov parameters) but also others uniformly in all of the settings of $(N_s, f_s) \in \{4, 8\} \times \{1, 50\}$. This demonstrates the necessity (and the superiority) of the negative log-likelihood as the distance in the context of our Markov model (kNMC) since the typical use of the Markov parameters as features is uniformly outperformed.

We next investigate certain other features that are commonly used in the literature. For instance, the authors of [24] study the social media application traffic and recommend the mean and standard deviation of the observations related to the rate signal (or equivalently the packet size), number of packets, inter-arrival time, port number and IP address as features to be used in the aforementioned compared algorithm (SVM, MLP, etc). We do not use the port number and IP address in our analyses, as they are not included in our introduced dataset; instead, we consider the rate signal, number of packets and inter-arrival time. On the other hand, the

Table 5.3 Overall average multiclass classification accuracy (%) at the application level for various state-of-the-art classifiers, using the statistics of packet size, number of packets and inter-arrival time

Classifier	**kNN**	**SVM**	**MLP**	**NB**	**RF**	**DT**
Accuracy	61.37 ± 6.50 (76.48 ± 3.45)	61.13 ± 6.87 (81.92 ± 6.16)	59.82 ± 5.43 (77.10 ± 8.20)	39.31 ± 2.92 (56.50 ± 5.92)	64.97 ± 5.98 (78.66 ± 5.19)	60.04 ± 4.04 (73.54 ± 5.10)

The accuracy result at the top in each cell is for the two-feature case, and the other below in parentheses is for the six-feature.

proposed DTMC modeling is, by design, applied to the rate signal (the aggregated packet size per unit time) in the presented work; hence, using features such as the number of packets and inter-arrival time that are not derivable from the rate signal is essentially using extra information that our classifiers do not exploit. For this reason, to ensure fairness, we focus on two sets of features: 1) the two features, i.e., the mean and standard deviation of the rate observations (packet sizes) within a window of traffic flow, and 2) all six features, i.e., the mean and standard deviation of the rate (packet sizes), number of packet and inter-arrival time observations. Here, only the case of two features provides a fair comparison between our algorithms and the state-of-the-art.

Table 5.3 compares the state-of-the-art classifiers, at the level of applications, among themselves when used with the above feature sets of size two and six. Accuracy figures of two features are written in this table explicitly and that of six features are written (below the one with two features in each cell) in parentheses. SVM and random forest outperform the others (kNN, MLP, naive Bayes, decision tree), whereas our kNMC with 85.71% accuracy in Table 5.2 (with $N_s = 8$ and $f_s = 50$) strongly outperforms SVM and random forest with two features, and still outperforms the both with six features even when they exploit the extra information of the number of packets and inter-arrival time which is not available to our classifiers. Hence, the results in Table 5.2 and Table 5.3 demonstrate the efficacy of the proposed Markov model in conjunction with the negative log-likelihood as the distance. In the rest of our experiments, we continue the comparisons with only kNMC, SVM and random forest (RF) as these three are observed to be the best performing.

Recall that our proposed kNMC is based on a Markov modeling of the quantized rate signal of the network traffic data, whereas the competing algorithms such as SVM and random forest (RF) are based on hand-crafted features such as the mean and standard deviation of the rate signal. Another powerful approach is to learn the features directly in a data driven manner rather than manually designing them or extracting from a model. A prominent technique of this feature learning approach is the autoencoder from the deep learning literature [61]. For this reason, we also include an autoencoder (AE) in the set of the methods that we compare with.

In our AE design, we use the raw traffic rate signal as a direct input to AE, after normalizing the rate to the interval $[0,1]$ and extracting the windows. This leads to an input data matrix of size $(N_{wc}N_c) \times (W_l f_s)$. We train AE based on the complete available data (without using the training and testing splits) since it does not use the label information (i.e. since it is unsupervised). Here, $N_{wc} \simeq 300, N_c = 7$ and $W_l = 120$ are all parameters of our training strategy and therefore they are all fixed across all of the compared algorithms. On the other hand, $f_s = 10$ Hz has been observed to perform better than $f_s = 50$ in this case (autoencoder) because increasing the sampling frequency directly increases the parameter complexity of the network. The compression rate from one layer to the next is 1/2 in our AE design with a bottleneck size 40 (note the dimensionality reduction from 1200 to 40), which leads to the use of 6 layers (we do not use compression only for the first layer) for the encoder. The decoder part is symmetrically defined from the bottleneck to the output (of the size of the input). We have also incorporated dropout (with probability 0.3) layers after each encoder layer, and used ReLu activation. This AE has been trained based on the reconstruction loss by using the Adam optimizer (learning rate: 0.001, batch size: 32, and 30 epochs without early stopping). After training, we detach the decoder and use the encoder as a feature extractor. In the last stage, based on the features provided by the encoder, we use random forest (RF) as the classifier because it generally performs better than the SVM. In this last stage, as the training is now certainly supervised, we repeat the same experimental procedure (such as training and testing splits) that we have previously explained for RF above.

Accordingly, Table 5.4 expands our results into the category as well as the application levels for these algorithms. We start with comparisons to SVM and random forest. The proposed kNMC is observed to outperform SVM and random forest at the category level with the overall multiclass classification accuracy 89.11% (against the best observed 87.41% with random forest of all six features and 79.91% with random forest of two features), in addition to its previously observed superiority at the application level with the classification accuracy 85.71% (against the best observed 81.92% with SVM of all six features and 64.92% with random forest of two features). In particular, the proposed kNMC is significantly more accurate for each individual category with an exception for S&MS, and this is also true for each individual application as well with an exception for “Whatsapp" which seems to be confused with “Spotify" (in the same category with “Whatsapp") more often compared to the other algorithms. It is also worth noting that our proposed kNMC achieves this superior performance when compared to the results of SVM and random forest for the two-feature case (only using the information of the rate

Table 5.4 Average classification accuracy ± standard deviations (%) per each application and category for kNMC, SVM, RF and AE+RF classifiers

Category	Classifiers				Application	Classifiers			
	kNMC	SVM	RF	AE+RF		kNMC	SVM	RF	AE+RF
VOD	93.45 ± 10.47	82.24 ± 9.28 (82.24 ± 8.16)	77.75 ± 7.25 (84.48 ± 6.26)	75.90 ± 12.19	Netflix	95.17 ± 8.48	58.96 ± 11.27 (77.93 ± 9.40)	58.27 ± 11.16 (62.06 ± 14.30)	74.00 ± 12.53
					Youtube	90.69 ± 14.36	57.58 ± 19.38 (77.93 ± 15.66)	54.13 ± 9.38 (73.10 ± 10.20)	47.79 ± 10.46
VLS	95.69 ± 4.47	75.68 ± 11.13 (86.20 ± 10.37)	76.89 ± 9.12 (81.72 ± 10.57)	86.44 ± 11.65	Youtube Live	96.55 ± 7.45	67.24 ± 21.27 (81.03 ± 17.05)	65.17 ± 18.40 (73.44 ± 16.46)	71.93 ± 15.03
					Twitch	94.48 ± 8.63	70.34 ± 16.11 (82.75 ± 13.44)	72.75 ± 17.68 (74.08 ± 22.09)	71.59 ± 29.35
S&MS	79.31 ± 7.54	74.65 ± 10.48 (86.03 ± 7.65)	81.89 ± 7.72 (90.34 ± 4.76)	87.79 ± 7.36	Spotify	67.24 ± 13.83	4.82 ± 5.16 (72.06 ± 15.30)	55.51 ± 11.27 (78.62 ± 7.83)	26.07 ± 26.45
					Whatsapp	68.97 ± 10.53	82.06 ± 13.51 (88.96 ± 12.12)	65.51 ± 16.89 (95.86 ± 9.22)	71.59 ± 29.35
TC	86.90 ± 23.38	85.51 ± 16.55 (92.75 ± 9.18)	83.10 ± 16.85 (93.10 ± 9.87)	95.72 ± 12.64	Skype	86.90 ± 23.38	85.51 ± 16.55 (92.75 ± 9.18)	83.10 ± 16.85 (93.10 ± 9.87)	95.72 ± 12.64
Overall	89.11 ± 4.01	79.52 ± 7.05 (86.81 ± 4.77)	79.91 ± 6.00 (87.41 ± 5.17)	86.46 ± 5.12	Overall	85.71 ± 4.08	60.93 ± 6.80 (81.92 ± 6.16)	64.92 ± 5.37 (78.66 ± 5.19)	65.53 ± 4.72

The accuracy result at the top in each corresponding cell is for the two-feature case, and the other below in parentheses is for the six-feature. The kNMC is presented with $N_s = 8$, $f_s = 50$ Hz, $k = 5$.

Table 5.5 Average classification accuracy per each application for KNMC-3D, KNMC, SVM, Random Forest and AE+RF classifiers on our dataset. kNMC-3D is presented with $N_s = 12$, $f_s = 50$ Hz, $k = 3$. kNMC is presented with $N_s = 8$, $f_s = 50$ Hz, $k = 5$.

App	Classifiers				
	kNMC-3D	***kNMC***	***SVM***	***RF***	***AE+RF***
Netflix	92.07	95.17	77.93	62.06	74.00
Youtube	88.28	90.69	77.93	73.10	47.79
YT Live	92.41	96.55	81.03	73.44	71.93
Twitch	93.79	94.48	82.75	74.08	71.59
Spotify	65.52	67.24	72.06	78.62	26.07
Whatsapp	72.41	68.97	82.75	82.75	82.75
Skype	90.00	86.90	92.75	93.10	95.72
Overall	84.93	85.71	81.92	78.66	65.53

signal). However, kNMC is still superior in 4 out of 7 applications when compared to the results of other classifiers for the six-feature case where the other classifiers exploit additional information of the number of packets and inter-arrival time. As for the comparisons to AE+RF (autoencoder followed by random forest), our proposed algorithm kNMC is significantly superior both at the category (kNMC: 89.11 vs AE+RF: 86.46) and application (kNMC: 85.71 vs AE+RF: 65.53) levels in terms of the average classification accuracy. Although AE+RF performs reasonably well at the category level, it suffers from within category confusions that largely degrade its application level performance.

Until this point, the performance analysis of kNMC is made in detail. Next findings mostly focus on kNMC-3D results. kNMC-3D is compared with kNMC and aforementioned benchmark methods that work with six features. The ones that employ

Table 5.6 Average classification accuracy per each category for KNMC-3D, KNMC, SVM, Random Forest and AE+RF classifiers on our dataset. kNMC-3D is presented with $N_s = 12$, $f_s = 50$ Hz, $k = 3$. kNMC is presented with $N_s = 8$, $f_s = 50$ Hz, $k = 5$.

	Classifiers				
Category	***kNMC-3D***	***kNMC***	***SVM***	***RF***	***AE+RF***
VOD	92.59	93.45	82.24	84.48	75.90
VLS	94.14	95.69	86.20	81.72	86.44
S&MS	87.24	79.31	86.03	90.34	87.79
TC	90.00	86.90	92.75	93.19	95.72
Overall	91.13	89.11	86.81	87.41	86.46

with two features are not included anymore because kNMC-3D also uses the information which is equal to six features. Table 5.5 reports the average classification accuracies per application on our dataset. According to the results in Table 5.5, While kNMC-3D performs almost as good as kNMC(0.8% worse), it is superior to other benchmark methods based on feature extraction. Although it is observed to have issues for classifying Whatsapp and Spotify, 84.93% overall average multiclass application accuracy is achieved, which is impressive. SVM, which is the best among the feature based state-of-the-art methods and the third best-performing method (after our methods), provides better precision on Whatsapp, Spotify and Skype against kNMC-3D. However, when we look into the accuracies of applications with video content, kNMC-3D achieves at least 10% higher accuracy than SVM per each application.

Accuracy results of the five classifiers on our dataset in terms of the categories are presented in Table 5.6. VOD category includes Netflix and Youtube; VLS category includes Twitch and Youtube Live; S&MS category includes Spotify and Whatsapp and lastly, TC category includes Skype. In general, category-based accuracies are higher than application-based accuracies as expected since applications of the same category could not be separated easily due to the similarities in traffic patterns. Our method kNMC-3D is the best-performing method among all five classifiers. Notably, almost all categories achieve classification accuracies over 90% with kNMC-3D. Other methods reach reasonable accuracies as well, but their performances are below that of kNMC. Especially in the recognition of VOD and VLS categories, kNMC-3D provides 8% higher performance against feature extraction methods. kNMC-3D gets ahead of kNMC in overall accuracy, thanks to its precision in S&MS category.

We start with comparisons to SVM and random forest. At the category level, our proposed algorithm kNMC outperforms SVM and random forest with the overall multiclass classification accuracy 98.24% (against the best observed 96.78% with

Table 5.7 Average classification accuracy ± standard deviations (%) per each application and category for kNMC, SVM, RF and AE+RF classifiers on the ISCXVPN2016 dataset

Category	Classifiers				Application	Classifiers			
	kNMC	SVM	RF	AE+RF		kNMC	SVM	RF	AE+RF
VOD	100 ± 0	80.00 ± 40.00 (100 ± 0)	80.40 ± 80.40 (100 ± 0)	40.00 ± 48.99	**Netflix**	100 ± 0.00	80.00 ± 40.00 (100 ± 0)	80.00 ± 40.00 (100 ± 0)	40.00 ± 48.99
S&MS	98.95 ± 2.22	34.44 ± 22.74 (100 ± 0)	93.33 ± 7.37 (97.78 ± 3.68)	95.54 ± 7.28	**Spotify**	70.00 ± 31.62	38.00 ± 32.80 (88.00 ± 29.93)	56.00 ± 30.72 (84.00 ± 21.54)	43.33 ± 23.80
					Skype File	78.33 ± 21.59	20.00 ± 21.05 (82.73 ± 27.10)	75.45 ± 19.94 (75.45 ± 25.08)	69.17 ± 21.10
					Facebook Chat	60.00 ± 51.64	NA (70.00 ± 45.82)	50.00 ± 50.00 (60.00 ± 48.99)	55.00 ± 41.53
					Hangouts Chat	70.00 ± 48.30	NA (100 ± 0.00)	40.00 ± 48.99 (70.00 ± 45.82)	50.00 ± 31.62
TC	94.00 ± 12.65	84.44 ± 13.33 (87.78 ± 17.53)	86.67 ± 13.88 (84.44 ± 18.72)	87.50 ± 11.93	**Facebook Video**	90.00 ± 31.62	70.00 ± 45.82 (80.00 ± 40.00)	80.00 ± 40.00 (60.00 ± 48.99)	80.00 ± 33.17
					Hangouts Video	94.00 ± 9.66	50.00 ± 46.10 (92.50 ± 16.01)	80.00 ± 26.92 (82.50 ± 31.72)	80.00 ± 32.25
					Skype Video	82.50 ± 31.29	52.50 ± 34.37 (70.00 ± 38.40)	67.50 ± 29.68 (77.50 ± 23.58)	80.00 ± 17.89
VO	100.00 ± 0	99.33 ± 2.00 (99.33 ± 1.42)	97.55 ± 5.48 (99.78 ± 0.67)	100.00 ± 0	**Facebook Audio**	67.50 ± 32.97	NA (75.00 ± 39.44)	81.67 ± 28.58 (94.17 ± 14.93)	92.50 ± 15.12
					Hangouts Audio	77.27 ± 19.76	99.09 ± 2.72 (80.45 ± 23.53)	93.18 ± 9.15 (98.18 ± 3.01)	96.82 ± 5.77
					Skype Audio	72.73 ± 28.75	NA (86.36 ± 14.22)	76.36 ± 19.58 (94.54 ± 11.64)	90.83 ± 15.11
Overall	98.24 ± 3.19	74.55 ± 11.30 (96.78 ± 4.46)	89.39 ± 10.79 (95.5 ± 4.85)	80.51 ± 12.12	**Overall**	78.39 ± 8.60	37.23 ± 5.74 (84.09 ± 8.33)	70.92 ± 10.65 (81.49 ± 8.41)	70.69 ± 10.27

The accuracy result at the top in each corresponding cell is for the two-feature case, and the other below in parentheses is for the six-feature. The kNMC is presented with $N_s = 10$, $f_s = 50$ Hz, $k = 7$.

SVM with all six features and 89.39% with random forest with two features). Out of 11 eleven cases at the application level, and when two features are used, our proposed algorithm kNMC outperforms the better one of SVM and random forest in 8 cases, performs comparably with the better one of them in 1 cases and underperforms in 2 cases ("Facebook Audio" and "Hangouts Audio"). On the other hand, when all six features are used, kNMC takes either the first or second place in 7 out of 11 cases. Here, we ignore small differences of a couple percents since the standard deviations for this ISCXVPN dataset are relatively large due to it (ISCXVPN) being ten times smaller than our introduced dataset. The overall average multiclass classification accuracy of kNMC at the application level is 78.39% which is approximately 7.5% higher than the better one of SVM and random forest with 2 features and approximately 5.5% lower than the better one of SVM and random forest with 6 features. Note also that four cases for SVM are labeled as not applicable (NA) as they define failures for SVM where the accuracy is around 0 due to the severe class imbalance in this dataset in terms of the sizes. As for the comparisons to AE+RF (autoencoder followed by random forest), our proposed algorithm kNMC is significantly superior both at the category (kNMC: 98.24 vs AE+RF: 80.51) and application (kNMC: 78.39 vs AE+RF: 70.69) levels in terms of the average classification accuracy.

Table 5.8 Confusion at the application level on the ISCXVPN2016 dataset (Classifier: kNMC)

Act \ Pred	Netflix	Spotify	Skype File	Facebook Chat	Hangouts Chat	Facebook Video	Hangouts Video	Skype Video	Facebook Audio	Hangouts Audio	Skype Audio
Netflix	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spotify	4.00	70.00	26.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Skype File	0.00	17.50	78.33	2.50	1.67	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Chat	0.00	0.00	20.00	60.00	20.00	0.00	0.00	0.00	0.00	0.00	0.00
Hangouts Chat	0.00	0.00	0.00	30.00	70.00	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Video	0.00	0.00	0.00	0.00	0.00	90.00	10.00	0.00	0.00	0.00	0.00
Hangouts Video	0.00	0.00	0.00	0.00	2.00	0.00	94.00	4.00	0.00	0.00	0.00
Skype Video	0.00	0.00	0.00	0.00	0.00	0.00	5.00	82.50	10.00	0.00	2.50
Facebook Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	67.50	22.50	10.00
Hangouts Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	15.91	77.27	6.82
Skype Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.18	19.09	72.73

Act: Actual, Pred: Prediction

Table 5.9 Confusion at the category level on the ISCXVPN2016 dataset (Classifier: kNMC)

		Prediction			
		VOD	S&MS	TC	VO
Actual	VOD	100	0.00	0.00	0.00
	S&MS	1.05	98.95	0.00	0.00
	TC	0.00	1.00	94.00	5.00
	VO	0.00	0.00	0.00	100

Therefore, on the benchmark dataset ISCXVPN [60], the proposed algorithm kNMC outperforms the state-of-the-art (SVM and RF) both at the category and application level for the two-feature case, but achieves a lower performance at the application level for the six-feature case only. The reason for this lower performance at the application level is the three applications ("Facebok Audio", "Hangouts Audio" and "Skype Audio") in which kNMC seems to fail. Since these three applications are confused only with the applications of the same category VO (the corresponding confusion matrices for the application and category levels are given in Table 5.8 and Table 5.9, respectively), the lower application level performance can easily be corrected at the VO category level reaching up to 100% accuracy as demonstrated in the leftmost column of Table 5.7. On the other hand, we attribute the confusion within the VO category to the Lloyd-Max quantization of the rate signal in the phase of determining the Markov states which is typically sensitive to outliers and class imbalance. We consider that a clustering algorithm (such as the expectation-maximization clustering) that takes into account the class priors and covariance structure can be helpful to remove that confusion. We also stress that the only fair comparison here is between kNMC and the two-feature classifiers (the mean and

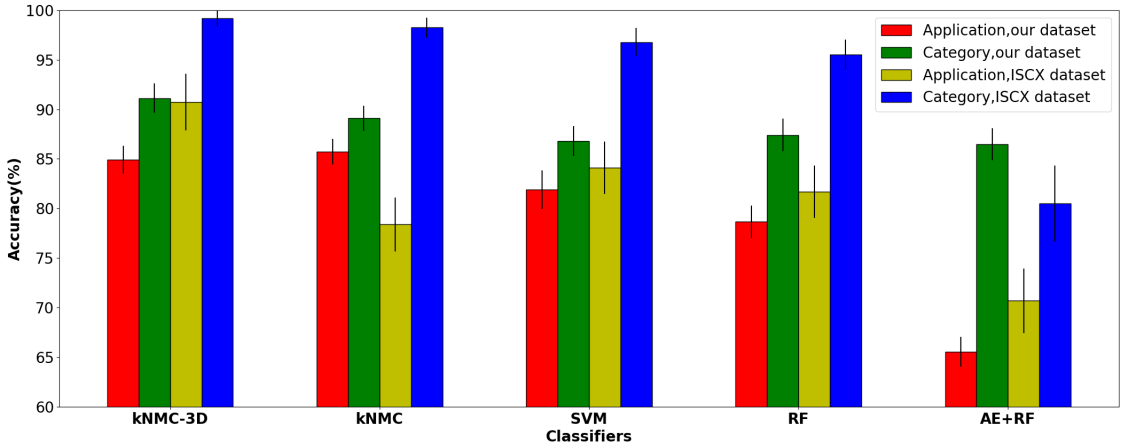

Figure 5.3 Overall average multiclass classification accuracy of the kNMC-3D classifier together with the benchmark methods is presented both application and category-wise on ISCXVPN and our dataset. kNMC-3D is presented with $N_s = 14$, $f_s = 50$ Hz, $k = 7$.

the standard deviation of the rate signal or packet size observations), where kNMC already outperforms SVM and random forest. We note that using features such as the number of packets and inter-arrival times that are not derivable from the rate signal provides an extra unfair advantage to SVM and random forest. This can be straightforwardly met with our Markov approach by a multi dimensional clustering of all the available observations (including packet numbers and inter-arrival times) while determining the Markov clusters (in other words, by using a voronoi cell vector quantization approach instead of a single dimensional Lloyd-Max quantization).

Overall, the proposed Markov modeling of the sequentiality (the statistical dependency across data instances) in multimedia traffic observations along with the corresponding classifier kNMC experimentally prove to be superior over the state-of-the-art feature extraction based approaches (SVM and RF) as well as the deep learning based alternative AE+RF at both the category and application levels in the case of the introduced dataset. This superiority is also observed at both the category and application levels in the case of the benchmark dataset ISCXVPN [60]. Note that the presented study mainly aims to demonstrate the efficiacy of using sequentiality with the proposed Markov modeling and hence we consider these extensions as future improvements. We also consider that the results with the introduced dataset are probably more reliable than that of the benchmark dataset as the former is a significantly larger (about 10×) dataset compared to the latter.

Considering all classifiers, the average multiclass classification accuracy values and their standard deviations, at application and category levels obtained on our dataset and the benchmark dataset ISXVPN are shown in Figure 5.3. In the ISXVPN

Table 5.10 Confusion at the application level on the ISCXVPN dataset (Classifier: kNMC-3D)

Act \ Pred	Netflix	Spotify	Skype File	Facebook Chat	Hangouts Chat	Facebook Video	Hangouts Video	Skype Video	Facebook Audio	Hangouts Audio	Skype Audio
Netflix	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spotify	4.00	84.00	9.38	2.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Skype File	1.58	7.42	84.55	6.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Chat	0.00	0.00	5.25	90.00	3.52	0.00	0.00	0.00	1.23	0.00	0.00
Hangouts Chat	0.00	0.00	5.69	2.45	90.00	1.86	0.00	0.00	0.00	0.00	0.00
Facebook Video	0.00	0.00	0.00	0.00	0.00	100.00	10.00	0.00	0.00	0.00	0.00
Hangouts Video	0.00	0.00	0.00	0.00	1.14	4.86	90.00	4.00	0.00	0.00	0.00
Skype Video	0.00	0.00	0.00	0.41	0.00	1.13	3.46	95.00	0.00	0.00	0.00
Facebook Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	75.83	20.12	4.05
Hangouts Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.24	98.18	0.58
Skype Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	13.60	85.45

Act: Actual, Pred: Prediction kNMC-3D is presented with $N_s = 14$, $f_s = 50$ Hz, $k = 7$.

dataset, our Markovian based kNMC-3D outperforms its competitors by achieving 90.73% overall application accuracy, which is at least 5% higher than its closest opponent, SVM. At the category level, kNMC-3D performs over 99%. In other words, almost all traffic categories could be separated perfectly with our method kNMC-3D. While kNMC falls behind other methods at the application level, kNMC-3D seems to eliminate the deficiencies of kNMC with multidimensional feature set. These results indicate that our method kNMC-3D has the potential to greatly improve QoS as classification is done almost perfectly at the category level.

More detailed results of the proposed method kNMC-3D on the ISCXVPN dataset can be seen in Table 5.10 which is a confusion matrix. We observe that our method has relatively high confusion in the audio cases. It predicts some of Facebook audio and Skype audio traffic as Hangouts audio. Therefore, per application accuracies of the two audio services are slightly low. Also, some of the applications in S&MS and TC categories are predicted falsely. However, these mistakes are mostly limited to in-category errors. For example, 9% of the Spotify traffic is predicted as Skype file. Although it affects the overall application accuracy undesirably, category based accuracies are not affected by these mistakes since Spotify and Skype file are from the same category.

Table 5.11 Average classification accuracy ± standard deviations (%) per each application for all proposed classifiers on our dataset

Applications	Classifiers						
	kNMC-3D	**OVA**	**OVO-SD**	**OVO-HD**	**CB-ECOC-SD**	**CB-ECOC-HD**	**2L-ECOC**
Netflix	92.07 ± 2.21	92.41 ± 10.99	92.41 ± 4.54	96.90 ± 7.14	100 ± 0	98.96 ± 1.58	100 ± 0
Youtube	88.28 ± 3.35	94.82 ± 10.25	94.83 ± 4.89	96.21 ± 6.44	100 ± 0	99.31 ± 2.07	100 ± 0
Youtube Live	92.41 ± 1.92	61.37 ± 12.02	80.18 ± 7.66	84.83 ± 13.19	100 ± 0	90.34 ± 7.03	100 ± 0
Twitch	93.79 ± 2.09	93.44 ± 10.28	90.16 ± 5.00	93.45 ± 8.65	84.14 ± 19.88	93.10 ± 15.11	84.14 ± 19.88
Spotify	65.52 ± 5.46	71.37 ± 14.79	79.19 ± 12.11	84.83 ± 15.51	100 ± 0	86.62 ± 18.75	100 ± 0
WhatsApp	72.41 ± 6.75	90.34 ± 18.87	87.27 ± 9.10	89.65 ± 13.17	100 ± 0	90.13 ± 21.54	100 ± 0
Skype	90.00 ± 3.01	88.62 ± 13.88	88.63 ± 8.07	88.96 ± 12.32	43.79 ± 18.89	88.27 ± 13.01	89.67 ± 4.33
Overall	84.93 ± 2.32	84.63 ± 5.19	87.60 ± 3.21	90.69 ± 4.04	89.70 ± 3.65	92.39 ± 4.10	96.26 ± 3.25

$f_s = 50$ Hz, $\{N_s,k\}$: kNMC-3D={12,3}, OVR={12,1}, OVO-Soft={12,3}, OVO-Hard={10,3}, ECOC-Soft={12,3}, ECOC-Hard={12,5}, Multi-level={12,3}

5.3 Performance of ECOC Based Classifiers

In this section, the accuracy performance of the problem-dependent unique ECOC classifiers are compared with base kNMC-3D and fixed problem-independent ECOC classifiers (explained in Chapter 4) by considering again, our dataset and ISCXVPN, employing the same train-test setups explained in previous section.

Table 5.11 reports per class average classification accuracies at the application level in our dataset. In terms of overall accuracies, ECOC-based classifiers generally outperform multiclass kNMC-3D. While CB-ECOC and OVO classifiers provide predictions with nearly 90% test accuracies, 2L-ECOC classifier is the best-performing one with 96% accuracy. Only OVA could not improve kNMC-3D. As explained in Chapter 4, sub-classifiers of OVA are trained like multiclass kNMC-3D because ECOC matrix is coded with 1 and -1's only and all data of the classes are included in training process. Therefore, improvement with the OVA is not expected. The kNMC-3D has low per-application accuracies in Whatsapp and Spotify. Although all ECOC-based classifiers can partially solve the differentiation problem of Whatsapp and Spotify; CB-ECOC-SD and 2L-ECOC classifiers resolve this issue completely. 2L-ECOC classifier predicts five classes without any mistake and reaches over 85% accuracy in the other two classes. Similar to the 2L-ECOC classifier, CB-ECOC-SD works without error in five applications; its overall application accuracy is slightly below the other ECOC-based classifiers due to the high error rate on Skype identification. As it can be seen in Table 4.1, it predicts a big proportion of Skype data as Youtube-live. OVO classifiers also have great performance when they are compared with multiclass kNMC-3D. OVO-HD has better performance than OVO-SD. Normally it is expected that classifiers with the soft decision logic are able to work better. However, in our case kNMC-3D-HD, also performs better than kNMC-3D-

Table 5.12 Average classification accuracy ± standard deviations (%) per each category for all proposed classifiers on our dataset

Applications	Classifiers						
	kNMC-3D	**OVR**	**OVO-SD**	**OVO-HD**	**CB-ECOC-SD**	**CB-ECOC-HD**	**2L-ECOC**
VOD	92.59 ± 1.39	94.14 ± 11.95	94.14 ± 3.22	98.45 ± 3.57	100 ± 0	96.55 ± 8.69	100 ± 0
VLS	94.14 ± 1.71	95.34 ± 6.41	94.84 ± 2.44	95.17 ± 5.39	100 ± 0	96.55 ± 6.54	100 ± 0
S&MS	87.24 ± 2.16	86.03 ± 7.08	88.97 ± 2.93	85.00 ± 9.25	100 ± 0	90.00 ± 5.04	100 ± 0
TC	90.00 ± 2.74	88.62 ± 13.88	88.62 ± 13.88	88.96 ± 12.32	43.79 ± 18.89	88.27 ± 13.01	89.67 ± 4.33
Overall	91.13 ± 1.12	91.03 ± 4.60	91.64 ± 1.60	91.90 ± 2.49	85.95 ± 4.72	92.84 ± 4.36	97.42 ± 1.92

$f_s = 50$ Hz, $\{N_s,k\}$: kNMC-3D=$\{12,3\}$, OVR=$\{12,1\}$, OVO-SD=$\{12,3\}$, OVO-HD=$\{10,3\}$, ECOC-SD=$\{12,3\}$, ECOC-HD=$\{12,5\}$, Multi-level=$\{12,3\}$

Soft. Therefore, the classifiers with hard decisions have superiority in general. The performance of OVO-HD is very close to CB-ECOC-HD, only $1-2\%$ worse. However CB-ECOC reaches that performance with only 12 sub-classifiers, while OVO classifiers make use of 21 sub-classifiers. It means that the OVO classifiers are nearly two times more complex than CB-ECOC classifiers. CB-ECOC-HD can classify five classes over 90% accuracy. while it has lower performance in Skype and Spotify identification.

Table 5.12 presents the per-category test accuracies of the ECOC-based classifiers. Similar the application case results, ECOC-based classifiers perform better than kNMC-3D. Especially 2L-ECOC reaches over 97% accuracy and becomes the best-performing classifier with at least $4-5\%$ higher performance. 2L-ECOC only fails at Skype identification and works on other categories without mistakes. The performance of the other classifiers are so close to each other. Problem-independent ECOC schemes and CB-ECOC-HD could not provide significant improvement to kNMC-3D. This is because if there is misprediction between categories, the similarity ratios are low and prediction becomes hard. Therefore, fixing out-category errors is challenging. The overall accuracy of the CB-ECOC-SD is again low compared to others. However, the problem is again Skype because TC category only includes Skype, and identification of Skype with CB-ECOC-SD fails at the application case. On the other hand, 2L-ECOC fixes CB-ECOC-SD errors. The main success of the CB-ECOC-SD is predicting categories with 100% success.(except for TC)

Average prediction accuracies of five classifiers at category and application levels on ISCXVPN dataset are represented in Figure 5.4. The results of OVO-SD and OVA are not reported for this dataset because their performances are lower than other classifiers. For better and clearer comparison, only five classifiers' results are shown. Surprisingly the performance of OVO-HD falls behind the performance of kNMC-3D. However other ECOC-based classifiers have better performances than kNMC-3D. In the benchmark dataset, 2L-ECOC classifier is again the best-performing classifier. At the application level, 2L-ECOC classifier provides over 95% accuracy.

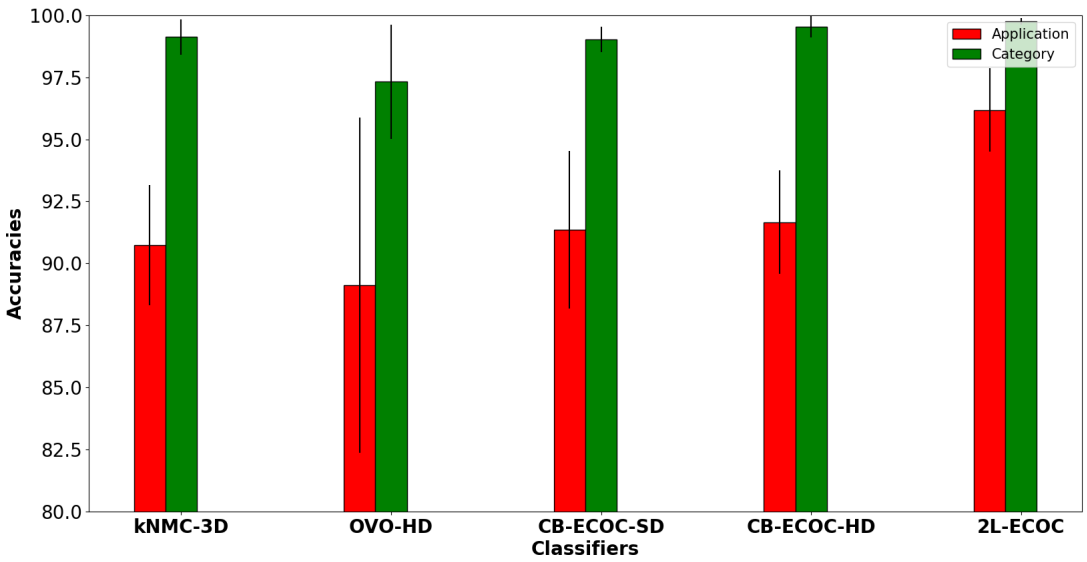

Figure 5.4 Overall average multiclass classification accuracies of the proposed classifiers for both application and category wise on the ISCXVPN2016 dataset

CB-ECOC-SD and CB-ECOC-HD classifiers perform nearly 3% below 2L-ECOC classifier at the application level. Their performances are very close to kNMC-3D but slightly better. In general, application accuracies in this dataset, are higher than the accuracies that are achieved in our dataset. CB-ECOC-SD fails to identify Facebook video application. As it can be seen in Table 5.15, Facebook video class is predicted with 10% accuracy and it is predicted as Hangouts video with 90% rate. The misprediction happens in the same category, therefore category accuracy is not affected by that failure. 2L-ECOC classifier solves the Facebook video, Hangouts video confusion in the second level. Therefore, the average application accuracy is higher.

5.4 Complexity Analysis

In this section, we present the computational and space complexities of the proposed classifiers, kNMC, kNMC-3D, CB-ECOC, 2L-ECOC, and also provide comparisons with the benchmark methods.

We start with the proposed classifiers kNMC, kNMC-3D, CB-ECOC and 2L-ECOC. First, sequential reading (i.e. not as a batch at once) of instances (for both training and test instances x_n and $x_{i,n}$) from an outer memory source is assumed; since

Table 5.13 Confusion at the application level on the ISCXVPN2016 dataset (Classifier: CB-ECOC-HD)

Act \ Pred	Netflix	Spotify	Skype File	Facebook Chat	Hangouts Chat	Facebook Video	Hangouts Video	Skype Video	Facebook Audio	Hangouts Audio	Skype Audio
Netflix	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spotify	4.00	70.00	26.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Skype File	00.00	17.50	78.33	2.50	1.67	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Chat	0.00	0.00	20.00	60.00	20.00	0.00	0.00	0.00	0.00	0.00	0.00
Hangouts Chat	0.00	0.00	0.00	30.00	70.00	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Video	0.00	0.00	0.00	0.00	0.00	90.00	10.00	0.00	0.00	0.00	0.00
Hangouts Video	0.00	0.00	0.00	0.00	2.00	0.00	94.00	4.00	0.00	0.00	0.00
Skype Video	0.00	0.00	0.00	0.00	0.00	0.00	5.00	82.50	10.00	0.00	2.50
Facebook Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	67.50	22.50	10.00
Hangouts Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	15.91	77.27	6.82
Skype Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.18	19.09	72.73

Act: Actual, Pred: Prediction

Table 5.14 Confusion at the application level on the ISCXVPN2016 dataset (Classifier: CB-ECOC-SD)

Act \ Pred	Netflix	Spotify	Skype File	Facebook Chat	Hangouts Chat	Facebook Video	Hangouts Video	Skype Video	Facebook Audio	Hangouts Audio	Skype Audio
Netflix	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spotify	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Skype File	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Chat	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hangouts Chat	0.00	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Video	10.0	0.00	0.00	0.00	0.00	10.0	80.0	0.00	0.00	0.00	0.00
Hangouts Video	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00
Skype Video	0.00	0.00	5.00	0.00	0.00	0.00	0.00	95.00	0.00	0.00	0.00
Facebook Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00	0.00
Hangouts Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00
Skype Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100

Act: Actual, Pred: Prediction

related computations can be performed in the online manner. The training phase for all these classifiers is common involving i) the parameter estimation for the DTMCs per each training instance $x_{i,n}$ of length $W_l f_s$, as well as ii) storing of the estimated parameters. However in the ECOC-based classifiers the parameters are estimated and are stored for each sub-classifier Note that the DTMC parameter estimations can be completed computationally highly efficiently and sequentially in a recursive online manner with $O(W_l f_s + N_s^2)$ complexity for a single training sequence at $O(N_s^2)$ space complexity due to storing the parameter matrix (N_s is the number of states in our DTMCs), cf. [40] for the details of recursive parameter estimations. Also, estimated parameters necessarily continue to be stored in the

Table 5.15 Confusion at the application level on the ISCXVPN2016 dataset (Classifier: 2L-ECOC)

Act \ Pred	Netflix	Spotify	Skype File	Facebook Chat	Hangouts Chat	Facebook Video	Hangouts Video	Skype Video	Facebook Audio	Hangouts Audio	Skype Audio
Netflix	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spotify	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Skype File	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Chat	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hangouts Chat	0.00	0.00	0.00	0.00	100	0.00	0.00	0.00	0.00	0.00	0.00
Facebook Video	10.0	0.00	0.00	0.00	0.00	66.5	23.5	0.00	0.00	0.00	0.00
Hangouts Video	0.00	0.00	0.00	0.00	0.00	0.00	96.40	3.6	0.00	0.00	0.00
Skype Video	0.00	0.00	5.00	0.00	0.00	0.00	0.00	95.00	0.00	0.00	0.00
Facebook Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00	0.00
Hangouts Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100	0.00
Skype Audio	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100

Act: Actual, Pred: Prediction

Table 5.16 Complexity analysis for the proposed classifiers in test time

Classifier	**Computational Complexity**	**Space Complexity**
kNMC	$O(N_{\text{tr}}(W_l f_s + N_s^2 + k + N_\alpha))$	$O(N_{\text{tr}} N_s^2 + k + N_\alpha)$
kNMC-3D	$O(N_{\text{tr}}(3W_l f_s + N_s^2 + k + N_\alpha))$	$O(N_{\text{tr}} N_s^2 + k + N_\alpha)$
CB-ECOC	$O((\sum_{i=1}^{N_c} N_{\text{tr,i}}(W_l f_s + k + N_\alpha)) + D)$	$O(\sum_{i=1}^{N_c} N_s^2 + k + N_\alpha)$
2L-ECOC	$O((\sum_{i=1}^{N_c+1} N_{\text{tr,i}}(W_l f_s + k + N_\alpha)) + D)$	$O(\sum_{i=1}^{N_c+1} N_s^2 + k + N_\alpha)$

test phase. Hence, overall, in the training phase, we have $O(N_{\text{tr}}(W_l f_s + N_s^2))$ computational complexity and $O(N_{\text{tr}} N_s^2)$ space complexity for kNMC and kNMC-3D, where N_{tr} is the number of training instances. For the ECOC classifiers, the training complexities of all sub-classifiers should be added up. Therefore for CB-ECOC, we have $O(\sum_{i=1}^{N_c} N_{\text{tr,i}}(W_l f_s + N_s^2))$ training complexity and $O(\sum_{i=1}^{N_c} N_{\text{tr,i}} N_s^2)$ space complexity where $N_c = 13$ is number of sub-classifiers. Since there is an additional sub-classifier in 2L-ECOC, the upper limit of the summation will be $N_c + 1 = 14$. There are different training sets for different sub-classifiers; so training for all the sub-classifiers are processed independently.

As for the test time, we analyze the computational and space complexity for each classifier separately for testing a single test instance x_n of length $W_l f_s$. Note that processing the test instance x_n for quantization and obtaining s_n are required for all the classifiers. This operation is done with different approaches in kNMC and kNMC-3D. We can say that the complexities of that conversion are nearly the same and negligible beside the classification part. For the classifier kNMC: there are

k-comparison/counting step (without a need for complete sorting for ease of exposition) and one additional majority step yielding $O(N_{\text{tr}}(W_l f_s + k + N_\alpha))$ (k is the number of neighbors as a parameter) computational complexity at $O(N_{\text{tr}} N_s^2 + k + N_\alpha)$ space complexity. For the classifier kNMC-3D: we can say that the complexity for single test entry is same as kNMC with the difference in the quantization part but it can be considered as negligible. Also, space complexity could be expressed similarly but the N_s will be bigger in kNMC-3D. For this reason, we can conclude that the overall complexity of kNMC-3D is slightly higher than the complexity of kNMC. Since the overall performance of kNMC is similar to kNMC-3D in our dataset, using kNMC seems more reasonable. However, the performance results in the benchmark dataset show that the performance of kNMC-3D is more solid with 10% higher accuracy. Hence, overall, kNMC-3D is preferable despite its relatively high complexity. The complexity of ECOC-based classifiers is calculated differently. The ECOC classifiers include N_c independent sub-classifiers thus the computational complexity will be a summation of all sub-classifiers' complexities. Also, ECOC-based classifiers have another additional complexity for decoding which is denoted by D but it is operated once, not for each sub-classifier. The complexity of ECOC classifiers is higher compared to kNMC-3D. But it should be noted that each classifier is independent and can be processed in parallel. Within parallel programming, the negative effect of high complexity can be removed. When the improvement in accuracy with ECOC classifiers is considered, the usage of ECOC-based classifiers could be a better strategy achieving better performance. However, those who need low complexity; can use kNMC with a small $W_l f_s$ parameter. Since 2L-ECOC has an additional classification level, its computational complexity will be higher than CB-ECOC. Also, the second level needs the predictions of the first level therefore they could not be run in parallel. We summarize these complexity findings in Table 5.16. As can be observed, the space complexities of ECOC classifiers are also higher because the state matrix for each sub-classifiers could be stored.

For the next phase, we compare the proposed methods with benchmark classifiers. One typically has computational complexity at least $O(N_{tr}^2 p_{tr})$ in training and $O(N_{sv} p_{tr})$ in test for kernel SVM, and $O(N_{tr} \log(N_{tr}) p_{tr} N_{tree})$ in training and $O(p_{tr} p_{depth} N_{tree})$ in test for random forest. Here, p_{tr} is the feature dimensionality, N_{sv} is the number of support vectors ($N_{sv} = N_{tr}$ in the worst case) and N_{tree} is the number of trees in random forest with p_{depth} being the maximum depth. Therefore, kNMC is computationally significantly more efficient than both SVM and random forest in training, since the complexity of kNMC scales linearly with respect to the training data size (N_{tr}) whereas the others scale with at least $O(N_{tr} \log(N_{tr}))$. In the test phase, we consider that kNMC and SVM are comparable complexity-wise,

since it would be not unreasonable to assume that the number of support vectors is not a negligible fraction of the number of training instances ($N_{sv} = N_{tr}$ in the worst case). Hence, the computational complexities of kNMC and SVM in test are both linear in the training data size (N_{tr}). On the other hand, random forest appears to be the computationally most efficient since its complexity does not depend on the training data size. However, there is one additional advantage for kNMC. We ignore here the complexity of the feature extraction phase that is necessary for SVM and random forest, whereas kNMC does not have such additional complexity as it directly sequentially operates on the traffic flow. This sequential operation is for Markov parameter estimation and can be performed fast (nevertheless, manifesting itself as the multiplication with $W_l f_s$ in our complexity figures). Finally, note that RF (with hand-crafted features) generally performs better than AE+RF whereas the computational complexity of extracting the hand-crafted features (such as mean and standard deviation of the rate signal) is essentially negligible compared to the training or running (one forward pass) an AE. Since kNMC and kNMC-3D are nearly the same in terms of complexity, all of the comparisons is also valid for kNMC-3D.

We know that ECOC classifiers are more complex than others, the complexity of CB-ECOC and 2L-ECOC are compared with the complexity of problem-independent ECOC schemes (OVO, OVA) that are presented in Chapter 4. The computational complexity of the problem-independent ECOC schemes can be expressed as $O(\sum_{i=1}^{N_c} N_{\mathrm{tr,i}}(W_l f_s + k + N_\alpha))$ where $N_c = 21$ for OVO case and $N_c = 7$ for OVA case. Here, i refers to index of sub-classifiers. The most important factor for that comparison is the number of sub-classifiers, N_c. Since OVO has more sub-classifiers than CB-ECOC and 2L-ECOC, the computational complexity of the OVO classifiers are higher. On the other hand, the OVA classifiers include less number of sub-classifiers and less computational complexity. However, the accuracies of OVA classifiers are relatively low compared to others. Although OVO classifiers have higher computational complexities, their performances are worse than CB-ECOC and 2L-ECOC. In terms of space complexities, the situation is similar. They are also proportional to the number of sub-classifiers.

6. CONCLUSIONS

In this thesis, we have considered multimedia traffic classification into popular applications (Youtube, Skype, etc.) and categories (video on demand, teleconferencing, etc.) in order to better serve the QoS requirements and enhance the user experience in Wi-Fi home networks. We formulate a multiclass classification problem in the Bayesian multi hypothesis detection framework and propose data-driven solutions based on a Markov modeling of the traffic source. To this end, four novel traffic classification schemes are presented.

The main logic of the classifiers relies on the conversion of packet-based traffic data to flow-based signals, which are processed by a sliding window to capture statistically stationary parts and classify timely. The windowed rate signal is quantized and modeled as a first-order discrete-time Markov chain (DTMC), providing an observation, i.e., instance, to the classification and the corresponding observation probability. Using the labels of training instances, each of which are also modeled as DTMC, the posterior class conditional probability is estimated as a mixture of Markov components. By considering the likelihood-based distance, k nearest neighbors of the test instances are determined for each test instance in kNMC, the main classifier of the thesis. Then, the kNMC classifier is redesigned with two more packet features: number of packets, and average inter-arrival times. Since the second classifier works with three packet features and applies the method of kNMC, it is named as kNMC-3D. While kNMC-3D performs as good as kNMC at the application level, it outperforms kNMC 91.78% accuracy at the category level in our dataset. In the benchmark dataset, kNMC-3D is superior to kNMC at both application and category levels.

To improve the DTMC-based classifiers, two custom-designed ECOC-based schemes are constructed with kNMC-3D: CB-ECOC and 2L-ECOC. The general design idea of the ECOC-based schemes is addressing the errors of kNMC-3D by investigating the confusion matrix. In overall accuracy, both ECOC classifiers outperform kNMC-3D at both category and application levels: CB-ECOC enhances kNMC-3D reaching 92.39% and 92.84% accuracy at the application and category levels respectively. 2L-

ECOC is the best-performing classifier providing 96.26% accuracy at the application level and 97.42% at the category level in our dataset. Similar results obtained with the benchmark dataset confirm the superiority of the proposed classifiers.

7. BIBLIOGRAPHY

[1] I. Capuni, N. Zhuri, and R. Dardha, "Timestream: Exploiting video streams for clock synchronization," *Ad Hoc Networks*, vol. 91, p. 101878, 2019.

[2] M. A. Hoque, H. Abbas, T. Li, Y. Li, P. Hui, and S. Tarkoma, "Barriers in seamless QoS for mobile applications," *arXiv preprint arXiv:1809.00659*, 2018.

[3] N. Roddav, K. Streit, G. D. Rodosek, and A. Pras, "On the usage of DSCP and ECN codepoints in internet backbone traffic traces for IPv4 and IPv6," in *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, 2019.

[4] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "WiFi can be the weakest link of round trip network latency in the wild," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, 2016.

[5] IEEE, "IEEE draft standard for information technology – telecommunications and information exchange between systems local and metropolitan area networks – specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) amendment 8: Medium access control (MAC) quality of service enhancements," *IEEE 802.11e, September 2005*, pp. 1–780, Dec 2005.

[6] L. Sanabria-Russo and B. Bellalta, "Traffic differentiation in dense collision-free wlans using csma/eca," *Ad Hoc Networks*, vol. 75-76, pp. 33 – 51, 2018.

[7] E. Khorov, A. Kiryanov, A. Lyakhov, and G. Bianchi, "A tutorial on IEEE 802.11ax high efficiency WLANs," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 197–216, 2019.

[8] L. D. Cicco, S. Mascolo, and V. Palmisano, "Qoe-driven resource allocation for massive video distribution," *Ad Hoc Networks*, vol. 89, pp. 170 – 176, 2019.

[9] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[10] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *IEEE Communications Surveys Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[11] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012.

[12] M. Welzl, S. Islam, R. Barik, S. Gjessing, and A. Elmokashfi, "Investigating the delay impact of the diffserv code point (DSCP)," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 612–616, 2019.

[13] S. Daoud and Y. Qu, "A comparison research on DSCP marking's impact to the QoS of VoIP-based and SS7-based phone calls," in *2019 7th International Conference on Information, Communication and Networks (ICICN)*, pp. 66–71, 2019.

[14] Y. Liu, G. Lu, W. Zhang, F. Cai, and Q. Kong, "A DSCP-based method of QoS class mapping between wlan and EPS network," in *Algorithms and Architectures for Parallel Processing*, pp. 204–213, Springer International Publishing, 2014.

[15] A. k. Jabbar, B. Karimi, T. M. Jamel, and A. Abood, "QoS Mapping Method Based on DSCP/IP in LTE and EDCA AC/MAC in WiFi Network," in *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*, pp. 662–667, 2019.

[16] R. Barik, M. Welzl, A. Elmokashfi, T. Dreibholz, S. Islam, and S. Gjessing, "On the utility of unregulated IP diffserv code point (DSCP) usage by end systems," *Performance Evaluation*, vol. 135, 2019.

[17] A. Custura, R. Secchi, and G. Fairhurst, "Exploring DSCP modification pathologies in the internet," *Computer Communications*, vol. 127, pp. 86 – 94, 2018.

[18] "Internet Assigned Numbers Authority (IANA)." http://www.iana.org/assignments/port-numbers. 2020-06-24.

[19] "Office 365 IP Address and URL web service." https://docs.microsoft.com/en-us/office365/enterprise/office-365-ip-web-service. Accessed: 2020-06-18.

[20] C. Xu, S. Chen, J. Su, S. M. Yiu, and L. C. K. Hui, "A survey on regular expression matching for deep packet inspection: Applications, algorithms, and hardware platforms," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2991–3029, 2016.

[21] A. Azab, R. Layton, M. Alazab, and P. Watters, "Skype traffic classification using cost sensitive algorithms," in *2013 Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 14–21, 2013.

[22] S. Galetto, P. Bottaro, C. Carrara, F. Secco, A. Guidolin, E. Targa, C. Narduzzi, and G. Giorgi, "Detection of video/audio streaming packet flows for non-intrusive qos/qoe monitoring," in *2017 IEEE International Workshop on Measurement and Networking (M N)*, pp. 1–6, 2017.

[23] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in *2017 International*

Conference on Computing, Networking and Communications (ICNC), pp. 843–848, 2017.

[24] F. Al-Obaidy, S. Momtahen, M. F. Hossain, and F. Mohammadi, "Encrypted traffic classification based ml for identifying different social media applications," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–5, 2019.

[25] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive IP traffic," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1880–1894, 2012.

[26] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[27] 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)*, pp. 439–454, IEEE, 2016.

[28] H. A. H. Ibrahima, S. M. Nora, and A. Ahmed, "Internet traffic classification algorithm based on hybrid classifiers to identify online games traffic," *Jurnal Teknologi (Sciences and Engineering)*, vol. 64, no. 3, pp. 55–60, 2013.

[29] H. A. H. Ibrahima, M. A. Al-Namari, and G. MohamedAli, "Internet traffic classification using machine learning approach: Datasets validation issues," in *Basic Sciences and Engineering Studies (SGCAC), 2016 Conference*, 2016.

[30] T. De Schepper, M. Camelo, J. Famaey, and S. Latre, "Traffic classification at the radio spectrum level using deep learning models trained with synthetic data," *International Journal of Network Management*, vol. n/a, no. n/a, p. e2100, 2020.

[31] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order markov chains and application attribute bigrams," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1830–1843, 2017.

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

[33] C. Zhang, X. Wang, F. Li, Q. He, and M. Huang, "Deep learning–based network application classification for sdn," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 5, p. e3302, 2018.

[34] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," tech. rep., 2013.

[35] A. Malik, R. de Fréin, M. Al-Zeyadi, and J. Andreu-Perez, "Intelligent sdn traffic classification using deep learning: Deep-sdn," in *2020 2nd International*

Conference on Computer Communication and the Internet (ICCCI), pp. 184–189, IEEE, 2020.

[36] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Know your big data trade-offs when classifying encrypted mobile traffic with deep learning," in *2019 Network traffic measurement and analysis conference (TMA)*, pp. 121–128, IEEE, 2019.

[37] B. E. Hansen, "Time series analysis," *Econometric Theory*, vol. 11, no. 3, pp. 625–630, 1995.

[38] B. Esmael, A. Arnaout, R. K. Fruhwirth, and G. Thonhauser, "Improving time series classification using hidden markov models," in *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pp. 502–507, 2012.

[39] J. Li, W. Pedrycz, and I. Jamal, "Multivariate time series anomaly detection: A framework of hidden markov models," *Applied Soft Computing*, vol. 60, pp. 229–240, 2017.

[40] H. Ozkan, F. Ozkan, and S. S. Kozat, "Online anomaly detection under markov statistics with controllable type-i error," *IEEE Transactions on Signal Processing*, vol. 64, no. 6, pp. 1435–1445, 2016.

[41] "Wireshark Documentation." https://www.wireshark.org/docs/. Accessed: 2020-06-21.

[42] H.Ozkan, R.Temelli, O.Gurbuz, and O.Koksal, "Multimedia traffic classification with mixture of markov components," *Ad Hoc Networks*, vol. 121, pp. 170 – 176, 2021.

[43] R. Darlington, "Dominance analysis: A new approach to the problem of relative importance of predictors in multiple regression," *Psychological Bulletin*, vol. 114, no. 3, pp. 542–551, 1993.

[44] J. Bring, "How to standardize regression coefficients," *The American Statistician*, vol. 48, no. 3, pp. 209–213, 1994.

[45] J. Johnson, "A heuristic method for estimating the relative weight of predictor variables in multiple regression," *Multivariate Behavioral Research*, vol. 35, pp. 1–19, 2000.

[46] A. Palczewska, J. Palczewski, R. Marchese Robinson, and D. Neagu, "Interpreting random forest classification models using a feature contribution method," *Springer International Publishing*, pp. 193–218, 2004.

[47] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[48] T. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes.," *Journal of Artificial Intelligence Research*, vol. 2, p. 263–286, 1995.

[49] S. Escalera, D. M. Tax, O. Pujol, and P. Radeva, "Subclass problem-dependent design for error-correcting output codes," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 30, no. 6, p. 1041–1053, 2008.

[50] O. Pujol, P. Radeva, and J. Vitria, "A heuristic method for application dependent design of ecoc," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 28, no. 6, p. 1007–1012, 2006.

[51] T. Hastie and R. Tibshirani, "Classification by pairwise grouping," in *Proc. Conf. Neural Information Processing Systems*, vol. 26, pp. 451–471, 1998.

[52] E. B. Kong and T. G. Dietterich, "Error correcting output coding corrects bias and variance," in *Inter-national Conference of MachineLearning*, pp. 313–321, 1995.

[53] E. Allwein, R. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers.," *Journal of Machine Learning Research*, no. 1, pp. 113–141, 2002.

[54] A. Berger, "Internet traffic classification algorithm based on hybrid classifiers to identify online games traffic," in *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 2006.

[55] T. Windeatt and G. Ardeshir, "Boosted ecoc ensembles for face recognition," in *2003 International Conference on Visual Information Engineering VIE 2003*, 2003.

[56] Z. YAN and Y. YANG, "Application of ecoc svms in remote sensing," in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-2, 2004.

[57] X. Xiao, Y. Bo, C. Yuehui, W. Lin, and C. Zhenxiang, "Network traffic classification based on error-correcting output codes and nn ensemble," in *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 475–479, 2009.

[58] J. Zhou, Y. Yang, M. Zhang, and H. Xing, "Constructing ecoc based on confusion matrix for multiclass learning problems.," *Science China Information Sciences*, no. 59, pp. 1–14, 2016.

[59] O. Pujol, S. Escalera, and P. Radeva, "An incremental node embedding technique for error correcting output codes.," *Pattern Recognition*, no. 41, pp. 713–725, 2008.

[60] A. Lashkari, G. Gil, M. Mamun, A. Ghorbani, and Y. Meng, "Characterization of encrypted and vpn traffic using time-related features," in *The International Conference on Information Systems Security and Privacy (ICISSP)*, 2016.

[61] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, "Deep learning," vol. 1, no. 2, 2016.