The Applications of Machine Learning Techniques in Networking

Area Exam
Soheil Jamshidi
Feb. 2019

***Abstract*—The growing complexity and scale of Internet systems coupled with the improved capabilities of Machine Learning (ML) methods in recent years have motivated researchers and practitioners to increasingly rely on these methods for data-driven design and analysis of wide range of problems in network systems such as detecting network attacks, performing resource management, or improving quality of service (QoS). In this survey, we review a large number of prominent research papers that apply ML methods to design, analysis or evaluation of network systems. To this end, we divide these studies into the following six groups based on the area of network systems that they target: 1) Domain name system, 2) Application identification, 3) QoS, 4) Cloud services, 5) Network security, and 6) Traffic prediction. Within each group, we examine the type of ML methods as well as input datasets that are used by individual studies, describe how they address various challenges, and summarize their key findings. In particular, we explore how domain knowledge in networking can inform different aspects of applying ML techniques such as problem formulation, feature engineering, feature selection, and deployment. We summarize representative common practices in the networking community along with a number of remaining challenges and research gaps.**

I. INTRODUCTION

During the past two decades, we have witnessed a significant increase in the scale and heterogeneity of network entities, underlying applications, and protocols. Both design and analysis of these protocols increasingly demand capturing and understanding patterns in large scale multi-dimensional datasets. The increase in network access speed, the appearance of bandwidth-hungry applications (such as video streaming, and P2P file sharing), the ISPs' increased interest in precise user traffic profiling to offer tailored services, and a response to the enormous growth in the number of connected users and IoT devices are among the main reasons for such a demand.
The early generation of networking studies have often relied on handcrafted, statistical techniques to identify desired patterns in different datasets solely based on known port numbers (*e.g.*, 21 for ftp, 80 for web), which was misleading in case of applications with a dynamic port assignment, such as P2P. Then, approaches evolve toward payload-based analysis, and due to constraints of encrypted communications, flow-level characteristics of the traffic was explored.

The scale of data and a higher level of the abstract in the flow-level datasets compare to port- and payload-based methods set the stage for ML methods. Recently, the prevalence of Machine Learning (ML) techniques with a proper fit for the mentioned challenges makes them a reasonable choice and led to growing deployments of these methods in design and evaluation of network systems.
This survey examines a body of recent studies that leverage various techniques in particular ML techniques for the design and characterization of network systems. Mainly, we group these studies by their target problem in networking that serves as a common context and background for them. Within each group, we further categorize based on more specific themes when possible. For each cluster of studies, we discuss the goal of each study and related challenges, its input dataset, the casting of their target problem into ML technique and associated challenges and issues, and their findings. Here, the focus is mostly on the formulation of the techniques for data analysis and related challenges, opportunities in particular how domain knowledge from networking has been used to customize any method. We select the well-received, highly cited, and peer-reviewed papers in top tiers networking venues such as IMC, SIGCOMM, TON, and InfoComm in the last ten years.
Instead of focusing on details of implementation, we review the research questions, challenges that are addressed, type of the ML methods and how they have been used (including features and approaches), and finally, the main findings. That will lead to a focus on how ML techniques are used to answer a specific question while ignoring less important details.
We categorize the prior studies into six sub-domains as follows:

1) **Domain Name System (DNS)**: We review studies that examine the ML-based solutions for challenges in performance and security of DNS. Namely detection of one-time-used domains to be excluded from caching [1] fluctuation of open DNS resolvers [2], and detection of malicious domains using decision tree models [3], and statistical techniques [4].
2) **Network Monitoring and Controlling**: We grouped studies in this section into two main sub-sections: protocol and application identification, and network controlling. Studies in this section, tackle various problems, such as the accuracy of labeled data [5], the role of different elements when applying ML techniques such as feature set or hyper-parameters [6], considering protocols as a language [7]. Detection of the zero-day (unknown) applications [8], real-time protocol identification [9] and considering only the packet length feature [10], as well as

the combined use of clustering and classification methods for traffic labeling [11], are among the novel studies reviewed in this section.

3) **Quality of Experience (QoE)**: The studies in this section mainly used the off-the-shelf ML techniques (mostly decision trees) leveraging the network traffic features for prediction and improving the user engagement and satisfaction in a range of applications such as video streaming [12], [13], [14], [15], Skype [16], and web [17], [18]. However, not all studies benefit directly from the ML techniques. Some studies get inspired by the ML techniques [15], some compare the results with ML techniques as a baseline [17], and some use them as an enabler to capture validated dataset [18].
4) **Security**: While most of the studies we review in all other sections have security implications, we group a number of them in this section that are solely addressing security concerns. User profiling, anomaly detection, and security or IoT devices are the common themes of these studies. Multiple ML methods including SVM, MLP, CNN, and DT are applied on user's (encrypted) web browsing data to group them with other similar users. Such grouping (user profiling) can be used for marketing and user segmentation purposes [19], [20], [21]. Anomaly detection methods have a wide range of flavors, such as detection of anomaly without prior context [22], statistical methods to detect anomalies [23] scaleable anomaly detection in mobile networks [24], detection focusing on temporary related traffic [25], anomaly detection based on server profiling [26], anomaly detection in the scale of WAN [27]. IoT related privacy challenges are summarized by Wang *et al.* [28]. This subsection follows by traffic analysis of IoT devices and related privacy and security concerns [29], and detection DDoS attacks in IoT devices [30] and behavior analysis of residential areas based on IoT traffics [31].
5) **Cloud and Cluster Computing**: In this section we review *HUYGENS* system [32] that explore the software clock synchronization in the network by predicting one-way propagation time, *Ernest* framework [33] that predicts the performance of large scale parallel analytic for efficient scheduling and resource management, and finally *Gaia* framework [34] that addresses the efficient usage of scarce WAN bandwidth in geo-distributed ML systems.
6) **Network Traffic Prediction**: In this section, we review studies in two groups: we review studies that leverage time-series analysis on network traffic data to forecast the traffic in the next time slot(s) along with studies with new ideas to obtain improved results. In the first group, studies focus on traffic prediction on mobile networks given its limited resources and facing increasing demand using Neural network methods [35] and unsupervised learning [36]. In addition, capturing spatio-temporal dynamics [37], using network traffic matrix [38], considering variations of LSTM [39], and predicting TCP output [40] are also included in this part. In the second group, improving LSTM by employing random connectivity trick [41], decomposing time-series [42], analyzing the prediction uncertainty factors [43], transforming the 1D traffic to 2D matrices and then applying the CNN [44], [45], encoding traffic to image [46], and utilizing statistical methods [44], [45], [47] are covered.

The rest of this survey is organized as follows: In section II we summarize the surveys that cover a similar topic. Then, in the next six sections, from section III to section VIII, we review the studies in each of the above-mentioned sub-domains. We discuss the practical challenges in section IX and conclude in section X by summarizing the studies and the pros and cons of different approaches along with open research questions that can be explored by the community.

II. Surveys in this domain

Surveys are considered as a good starting point for summarizing the problems, solutions, comparison factors, and different challenges in any domain. There are a few surveys on the application of machine learning methods in networking that we review in this section.

In a related survey that has been done in CAIDA, Zhang *et al.* [48] (Check out their interactive webpage [1]), they present a taxonomy that can be used in order to answer questions in this domain, such as *what fraction of traffic is P2P?*. To this end, they break down the *classification goals* into two main set: i) Coarse classification goals, i.e., whether it's transaction-oriented, bulk-transfer, or peer-to-peer file sharing, ii) Finer-grained classification goal, i.e., the exact application generating the traffic. They categorize the *methods* into 1) the exact matching, 2) heuristics, 3) supervised or 4) unsupervised ML methods. In Figure 1 they show the trends of applications and feature types used over time. They discuss

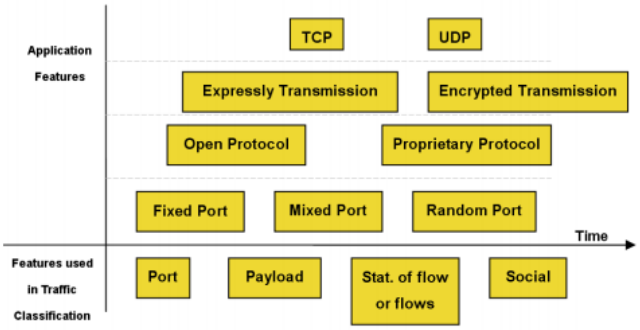

Fig. 1. Trends of applications and features by Zhang *et al.* [48]

the results in terms of annual trend, link, and location-based ranges for P2P connections, However, they mentioned that "*they have far too little data available to make conclusive claims beyond the general indications and insights per study*".

[1] http://www.caida.org/research/traffic-analysis/classification-overview/

In a recent effort, Ashraf *et al.* [49] focus on applications of ML techniques in Software Defined Networks (SDN). They review prior studies that leverage different ML methods to tackle related issues to SDNs. The main takeaway from this study is the comparison of ML methods used in all reviewed studies. Resource consumption, training time, and over-fitting are among the main shortcomings, while accuracy and ability to learn from sparse and noisy data considered to form a good classifier.

Wang *et al.* [50] provide a survey on different networking problems which have been addressed by machine learning approaches in the past, including objectives such as traffic prediction, traffic classification, network management, self-configuration, as well as performance analysis and prediction. They look at this as an inter-discipline domain and they try to make a practical guide for network researchers by providing the workflow of applying ML techniques and describe the recent advances. They focus on a small number (less than 15) of studies to showcase different aspects of the workflow instead of covering efforts that have been done in this domain.

In a recent study, Boutaba *et al.* [51] survey the evolution, application, and research opportunities of using machine learning in networking domain. They start with an overview of the machine learning methods, approaches, and sources to obtain data (network traffic), feature engineering techniques, and evaluation methods that are used in the reviewed studies. Then, they review the benefit of ML in traffic prediction, classification and routing, congestion control, resource management, fault management, and QoS/QoE management for networking, anomaly and misuse detection for intrusion detection in networking. In the end, they also discuss the importance of online learning, support for secure learning, and system architectural design to ease the use of ML approaches for networking. Covering more than 500 studies from different angles, they offer a uniquely comprehensive survey on this matter.

Section Summary: We reviewed surveys that are focused either on network traffic challenges, or applications of machine learning on network traffic. While there are recent and related studies on this matter, they are either so broad or specific on the topic that can not act as a concise yet complete guide for researchers in this domain.
In the following sections, we review the studies in each networking sub-domain to compare and contrast their contributions and summarizing the learned lessons.

III. Domain Name System (DNS)

The Domain Name System (DNS) provides mapping service for Internet users by translating domain names (*e.g.*, `www.uoregon.edu`) to IP addresses (*e.g.*, `128.223.142.244`). Serving the mapping request, AKA DNS lookup, is done in a globally distributed fashion, for which there is a hierarchy of DNS servers (resolvers) responding to client's requests (DNS queries). The scale and architecture of such a system expose it to a range of challenges, mainly *performance* and *security*. In this section, we will review studies that applied ML methods to address challenges related to DNS.

A. DNS performance

Given the large scale of DNS usage, different techniques such as caching are used to improve the performance of the system. In caching, the goal is to reduce the number of repetitive requests by storing (caching) the mapping result for frequent DNS lookups in the lower layers of the DNS hierarchy to save higher layers from overwhelming repetitive requests.
To this end, first, we need to know which DNS lookups are frequent. Hao *et al.* [1] propose an ML-based method to identify host names that are likely to be one-time-only, assuming that not caching these domain names can improve the performance caching in DNS. They test the DNS caching performance on actual DNS lookups that they passively collected from 2 large campus networks over two weeks. These datasets cover 4M (13M) unique domain names and 3 (10)M equal to 86% (76%) once-used names. Each includes attributes such as Query Name (N), Structure of Domain Name (S), Query Type (K), and Query Time-stamp (T). They cast the problem into a two-class (re-use, once-used) classification, where decision tree and random forest models (in `R`) are applied. To handle the unbalanced nature of classes in the dataset, they have assigned different weights to each of the classes. The set of features used in this study includes Length of the query name, Length of the longest subdomain name, sub-domain depth, Number of format fields, Number of fields with unusual lengths. Their results indicate that a caching policy that does not cache domain names that are labeled as one-time-only, exhibits 10% improvement compared to standard caching policies like Least Recently Used *LRU* and First In First Out *FIFO*.

B. DNS Security

Given the distributed nature of DNS, this system has been a target and at the same time a tool for many security threads. In this part, we review studies that focus on illegitimate responses to DNS lookups, constant domain name changes (Domain Fluxing), and how DNS queries can be used to identify benign and malicious traffics.

Open DNS Resolvers: DNS is composed of a large number of DNS servers that are willing to resolve recursive DNS lookups for anyone on the Internet, these servers are called "open DNS resolvers". Open resolvers can respond illegitimately and redirect users to unintended destinations for censorship, malicious, or marketing purposes. Kuhrer *et al.* [2] analyze the fluctuation of open DNS resolvers over time. They took the viewpoint of a client system and determined the response *authenticity* and *integrity* of malicious open DNS resolvers in IP4 address space.

They have done active probing by sending billions of DNS lookup requests for 155 domains to all open DNS resolvers in the entire IPv4 address space, over 13 months to collect an extensive dataset on open resolvers over time. Next, they try to pre-filter the legitimate resolvers to find a subset of resolvers that have malicious behavior. Then, for each (domain, IP, resolver) tuple they request an HTTP content to analyze the differences compared to the legitimate version. To reveal the type of response manipulation in open DNS resolvers with an illegitimate response, they use clustering techniques. They use unsupervised learning methods (agglomerative hierarchical clustering) in two steps. First, to lower the scale of HTTP payload data that they had to inspect manually, they applied a coarse-grained clustering to group similar HTTP responses. They apply agglomerative hierarchical clustering an unsupervised clustering method using a custom distance function to measure the similarity of two individual HTTP responses using seven normalized features:

- Length difference of the HTTP response body
- Jaccard distance for multisets with the set of HTML tags in the HTTP payload.
- Edit distance between the sequence of opening HTML tags
- Edit distance on the $< title >$ value
- Edit distance of all JavaScript code
- Jaccard distance for all embedded resources (i.e., the values of $src =$ "" attributes), and for outgoing links (i.e., the values of $href =$ "" attributes)

Then, a fine-grained clustering was applied to find page modifications using the Jaccard distance of the number of added or removed HTML tags compared to the ground truth version of HTML documents. As a result of the manual inspection, they assign labels to the groups to be blocking, censorship, login, parking, misc., HTTP error, and search. They reported that during their 13-month-long study, the number of DNS resolvers dropped from 26.8 to 17.8 million servers. They identified millions of resolvers that deliberately manipulated DNS resolutions and returned unexpected IP address information. Examples are a redirection to censor communication channels, inject advertisements, serve malicious files, or perform phishing. Requiring a mechanism for validation at the application layer to protect end hosts from manipulated responses.

Malicious Domain Names: Given the distributed nature of DNS servers; they have always been a target for attackers and malicious activities. For example, there is a technique for keeping malicious botnets in operation by constantly changing the domain name of the botnet owner's Command and Control (C&C) server, and it is called Domain Fluxing. Domain Fluxing not only increases the rate of DNS lookup requests sent out by each bot but makes it hard for network admins to block the botnet instructions and shot down the botnet. Bilge *et al.* [3] focus on detection of such malicious domains using machine learning methods in a scalable manner and proposed their framework called *EXPOSURE*. They collect a real-world dataset of more than 100 billion DNS queries to 4.8 million distinct domain names for two and a half months in North America and Europe. Their method was also deployed in an ISP to check its real-time applicability. DNS query features are used to classify domains as malicious or normal. They apply J48 Decision Tree (C4.5) as an efficient and accurate option using the time (life length, daily similarity, access ratio, and repeating pattern), DNS response related (#distinct IP addresses, #distinct countries, #domains with the same IP, reverse DNS result), TTL related features (average TTL, standard deviation of TTL, # distinct TTL values, #TTL changes), and domain name related (%numerical characters and % of length of longest meaningful substring) to classify malicious domains. They discuss the feasibility and accuracy of their models in identification of malicious domains.

This approach is similar to the study done by Yadav *et al.* [4] that develop a methodology to detect "Domain fluxes" in DNS traffic by looking for patterns inherent to domain names that are generated algorithmically, in contrast to those generated by humans. They leverage statistical measures such as *Kullback-Leibler divergence*, *Jaccard index*, and *Levenshtein edit distance* to decide whether a group of domains are malicious (or algorithmically generated) or not.

DNS as an Enabler: Given the wide usage of DNS on the Internet, DNS lookup queries can be used to collect more information for profiling the network entities, clients or servers. Fukuda *et al.* [52] suggest to use improper use of DNS as a new source of information to detect benign and malicious network-wide activities. A `network-wide` activity is when one computer (the originator) contacts many others (the targets). Motives for network-wide activity may be *benign* (mailing lists, CDNs, and research scanning), *malicious* (spammers and scanners for security vulnerabilities), or perhaps *indeterminate* (ad trackers). The knowledge of malicious activity may help to predict attacks, and understanding benign activity help to understand the current state of the system and characterize its future changes. Their approach is simple: collecting DNS data, classifying data for each originator with features based on the queries and queriers, and then clustering originators into activities. For their experiments, they leveraged passively collected datasets that are available for researchers through DNS-OARC. The goal here is to use the temporal and spatial information of DNS queries to classify the legitimacy of the traffic. They use a decision tree (Classification And Regression Tree; CART), random forest (RF), and support vector machines (SVM). Two sets of features are used: First, to capture the temporal characteristics they relied on: queries per querier and query persistence; Then, to capture the spatial characteristics, they focused on local and global entropy, unique ASes and countries, queriers per country and per AS. They classify queries into 12 benign and malicious classes

(multi-class) and discuss the accuracy of their models.

Section Summary: We reviewed studies that use machine learning techniques to address some challenges in the Domain Name Systems (DNS) mostly in recent years (after 2015). The common theme for studies in this domain is the assessment of integrity, authenticity, and fluctuation (security aspects) of DNS. The decision tree and random forest are mostly used as the main or one of their ML techniques. Mainly defined the problem as classification with two class labels (malicious and non-malicious or oneTimeUsedDNS and reUsedDNS). In multi-label scenarios, SVM classifier is also considered. Numerical features are the most common type of features. However, features are domain specific and vary by different studies. The main issue or weakness in this domain's studies is the lack of attention to *feature selection* approaches, reporting the computational time/costs, and *feature engineering* to craft more robust and aggregated features.

TABLE I
DT: DECISION TREE RF: RANDOM FOREST CL:CLUSTERING

	Hao[1]	Kuhrer[2]	Fukuda[52]
Real world dataset	✓	✓	✓
ML method	DT/RF	CL	DT/RF/SVM
Feature analysis	X	X	X

IV. NETWORK MONITORING AND CONTROLLING

The large scale, time-sensitive, and bandwidth-hungry applications that are deployed on the Internet, require innovative approaches for controlling and monitoring. Decision making on the priority of different traffic classes as well as observing the network's routing status are at the core of network monitoring. In this section, we cover the efforts that have leveraged the ML techniques to tackle network routing issues, identify protocols or applications, and classify the network traffic in order to offer high performance and Quality of Experience (QoE) to clients.

A. Protocol and Application Identification

The protocol identification reveals the type of protocol and application associated with each network flow. This enables we to assess behavior of network flows and allocate proper resources. As a result, network administrators can consider tailored resource management policies in order to improve the quality of service (QoS) for intended and high priority protocols.

Soysal *et al.* focus on the classification of Internet traffic protocol [5] by classifying the traffic into P2P (P), content delivery (C), web (W) (HTTP), bulk (B) (FTP), service (S) (DNS), and mail (M) (SMTP) using the Internet traffic flow traces. They apply three supervised machine learning methods: 1) Bayesian Networks (BN), 2) Decision Trees (DT), and 3) Multilayer Perceptrons (MLP) on a dataset acquired from the National Academic Network of Turkey (ULAKNET and ULAKBIM). The accuracy of labeled data is the main difference between this study and prior works. Instead of relying on labels gathered from the port- or payload-based inspection methods, they obtain the flows from servers that serve a distinct protocol service, so the traffic types of the flows originated from each server is known without ambiguity. They sample the traffic flows and use flow features such as protocol, source, and destination port, packet size, Type Of Service (TOS), and flags to keep the computational cost of feature engineering low. Using Weka implementations, they suggest that DT are the most suitable for traffic classification in terms of computational cost and accuracy, while the performance of machine learning methods with backpropagation was considerably lower than BN and DT. In addition, they test the effect of an unreliable port number in training set and explain how port-based labeling can lead to an inaccurate evaluation.

In an effort to propose an accurate traffic classification approach that considers real-time analysis, Michael *et al.* [6] apply the Neural Network (NN) on campus-wide bi-directional TCP/IP flows that are semantically complete, *i.e.*, completed the connection setup and observed tear-down. They use flow-level metrics (such as total bytes, inter-arrival times, total packets), metrics related to SACK, SYN, FIN, and URG packets, segment and window advertisement size, missed data, payload size, and RTT statistics as features. Data was manually labeled employing deep packet inspection (DPI) methods as explained by Moor *et al.* [53] and automated by Canini *et al.* [54]. Flows are divided into 10 groups, including Bulk (ftp), Database, Interactive (SSH, telnet), Mail, Services (imap, stmp), WWW, P2P, Attack (virus and worm attacks), Games (Microsoft direct play), and Multimedia (real and windows media player). Using a Multi-Layer Perceptron (MLP) model, they consider different activation functions (tanh, sigmoid, Relu), along with different set of attributes (performing feature selection), and assess the temporal- (data after a year) and spatial- (using the captured data from different parts of the campus) stability of accuracy. The shortcomings of the above approach are as follows: features should be defined per protocol and labeling quality depends on the quality of DPI tools. Therefore, their approach may not scale.

To avoid the need for prior knowledge of the protocol's specification. Yun *et al.* [7] propose *Securitas* as a network trace-based protocol identification system. Securitas exploits the semantic information in protocol message formats without relying on prior knowledge of protocol specifications. Their method considers each protocol as a language and leverages n-grams to find the most frequent (signature) of protocols, using LDA (Latent Dirichlet Allocation) and approximate inference for keyword inference and feature extraction. They use the following supervised ML techniques for testing: SVM,

C4.5 Decision Tree, and Bayes Network. To quantitatively evaluate the effectiveness of their system, they use millions of flows (75M and 156M flows) obtained from a backbone router of a major ISP in China that contain seven typical and stateful protocols, including BitTorrent, PPLive, SMTP, DNS, FTP, SIP, and CIFS/SMB. Their results suggest that this method have high accuracy on different types of protocols (connection-oriented and connectionless) as well as short and long-term, binary and textual flows.

In the presence of Zero-day (*i.e.*, unknown applications), prior knowledge about the application is often not available. Zhang *et al.* [8] propose a Robust Traffic Classification (RTC) framework to tackle the issue of zero-day applications for network traffic classification. In the 3-module design of their framework, they first find training samples of unknown applications from the unlabeled data using K-means clustering method. Then, using the new unknown class, they perform a (N+1)-class classification (Random Forest) on the data to label the known and unknown classes. They use a Bag of Flow approach [55] using a 3-tuple heuristics which applies the majority of labels as the label of all flows that have the same destination IP, port, and protocol. The last module of their framework is a system update that deals with the drift in the data and sub-classes that might exist in the unknown class. They report the performance results on four datasets collected from a campus, an ISP, and backbone links, discussing the F-measure, TPR, and FPR as well as classification of flows per second compared to the other methods. Their results demonstrate that RTC outperformed ML methods including: random forest, correlation-based classification, semi-supervised clustering, and one-class SVM.

In addition to the accuracy, performance of protocol identification methods is a critical determining factor in deployment of such methods. Specially, given the scale and time granularity that controlling policies should be applied. Santiago del Rio *et al.* [9] focus on the real-time protocol identification where they present a software-based traffic classification engine running on commodity multi-core hardware. Their proposed approach is able to process aggregates of up to 14.2 Mbps over a single 10 Gbps interface in real-time. That is the maximum possible packet rate over a 10 Gbps Ethernet links given the minimum frame size of 64 Bytes. They leverage the CAIDA's anonymized 2009 Internet traces for experiments. This significant advance in the classification rates are achieved by (i) leveraging an improved network driver, *PacketShader*, to efficiently move batches of packets from the Network Interface Controller (NIC) to the main CPU; (ii) deployments of lightweight statistical classification techniques (*i.e.*, Naive Bayes with Gaussian density estimation) to exploit the size of the first few packets of every observed flow; (iii) a careful tuning of critical parameters of the hardware environment and the software application itself. Although they utilize a custom implementation of Naive Bayes with Gaussian density estimation for their experiments, they propose to consider J4.5 Decision tree since it is more efficient for online learning.

Yamansavascilar *et al.* [56] explore the accuracy of different methods such as J48, Random forest, k-NN, and Bayes Network application identification. Experiments rely on a public (UNB ISCX Network Traffic - 15K flows) and an internal (crawled to cover social media and music streaming services - 4K flows) dataset. They show that k-NN ($k = 1$) and random forest are the most accurate predictors. Having 14 different applications organized in a number of groups (File transfer, Mail, P2P, Streaming, Instant Messaging, and Social media), they use minimum, maximum, and average packet size, and flow count as well as window and byte related features (minimum, maximum, and average window size). The reported accuracy on these two datasets is 94% for the public dataset and 91% for the internal dataset.

Among the long list of features used for application classification in the prior studies, Wang *et al.* [10] consider only the packet length to classify the applications using a decision tree. They assume that the sequence of network packets in a flow offer a unique signature that can identify application layer protocols, and they prove it by experiment. Their dataset is provided by an ISP, that they label it relying on an off the shelf program called *Proto_ident*. Proto_ident finds application protocols according to the first four bytes of the packet payload. Application of decision trees are prevalent in this domain due to the simplicity of presentation and also the ability to tolerating the noise. In this study, the J48 tree, RandomForest, and REPTree classifiers of the *Weka* framework are used. Overall, 98% accuracy is reported by this study across all classifiers.

In a visionary paper, a fine-grained mobile application detection framework (called *Atlas*) is proposed by Qazi *et al.* [57]. *Atlas* incorporates application awareness into Software-Defined Networking (SDN). It is capable of L2/3/4-based policy enforcement but agnostic to higher layers. Atlas enables fine-grained, accurate and scalable application classification in SDN. It employs open source c5.0 decision tree method, a crowd-sourcing approach to obtain label for the data, and leverages SDN's data reporting mechanism and centralized control. They prototype Atlas on HP Labs wireless network. Based on the first N packet sizes, port numbers, IP address range as features, they observe 94% accuracy on average, for top 40 Android applications.

The combined application of clustering and classification methods for traffic labeling is suggested by Szabo *et al.* [11] as a hybrid method. However, the implications of the processing time are not discussed. They propose a framework for traffic classification that employs machine learning techniques and uses only packet header information. First, a combination of clustering and classification algorithms are used to obtain robustness when network parameters changes, *e.g.*, when

the test and train data are collected from different networks. Second, they explore multiple traffic granularity levels and propagated information between the levels to increase accuracy and accelerate classification. Third, they customize a set of constraints based on connection patterns to utilize state-of-the-art clustering algorithms efficiently. Leveraging the K-means clustering method, they group up the related flows per application and label the flows. Then, they train one classifier per cluster by C5.0 decision tree. They use one day measurements from 2G and 3G networks in Asia, Europe, and North America to have variation in type and geo-location of their data. Individual flows are defined as bi-directional with a 5-tuples identifier (i.e., protocol, source IP, source port, destination IP, and destination port) and a one minute timeout. Flows are labeled with a DPI tool and are randomly divided into the training and testing. Selection is done with 1/100 probability from those flows where the protocol is recognized by the DPI tool and contained at least 3 packets.

Zander *et al.* [58] utilize the statistical characteristics of the flows in order to classify their corresponding application and protocol. They use an unsupervised Bayesian classifier (called *autoclass*) that automatically learns the "natural" classes (clusters) inherent in a training dataset with unclassified instances. They evaluate the efficiency of their approach using data from several traffic traces collected at different locations of the Internet. Furthermore, they perform feature selection to identify an optimal feature set and determine the influence of different features on random flows with more than 3 packets (1k flows per application). The feature selection step was done using sequential forward selection (SFS) to find the best attribute set because an exhaustive search is not feasible. They examine the feature importance and show that forward packet length variance is the most important one, following by backward packet length variance and packet bytes. Then, the resulting classifier can be used to classify new unseen instances. They use three sets of datasets: *Auckland-VI*, *NZIXII* and *Leipzig-II* traces from The National Laboratory for Advanced Network Research (*NLANR*).

Application and protocol identification is not limited to the classification of known groups. Glatz *et al.* [59] study the classification of one-way flows that contain packets only in one direction. They show that one-way flows form between 34% and 67% of the total number of flows. However, such flows account for only 3.4% and 0.79% of the total number of packets and bytes, respectively. They focused on the following classes:

- **Benign P2P**: P2P applications are trying to access peers listed in their local host cache that are not available anymore.
- **Backscatter**: replies to DoS attack traffic that uses randomly chosen source IP addresses to hide the real identity of an attacker.
- **Suspected Benign**: one-way flows may exist as a part of benign applications using data and control connections in parallel for acknowledgment only. Another cause may be a temporary failure within an otherwise valid communication.
- **Bogon**: one-way flows originating from bogon IP space.
- **Other**: one-way flows that do not match any of the above classes.

They leverage signs (features) based on host pair, remote, and local host behavior as well as flow features (*e.g.*, protocol, # packets, single packet flow). They use the decision tree and report its inferred rules, which is an aspect seen in multiple prior studies.

To understand how Internet service providers can benefit from deploying ML approaches, Pietrzyk *et al.* [60] adopt the perspective of an ADSL provider. Given that it is too costly to deploy a Deep Packet Inspection (DPI) tool at each point of presence (PoP) of an ISP, they use a statistical classification as a complementary tool to DPI. A typical use could be to devise a statistical classifier built upon the knowledge (reference point) collected on the PoPs where some DPI solutions are available, to be deployed where those DPI solutions are missing. They focus on supervised statistical classification where a specific ML algorithm is trained on the training set (for which the reference point is known), using a specific set of features. Their dataset consists of four recent packet traces collected at three different ADSL PoPs from the same ISP in France. Their probes are located behind a Broadband Access Server (BAS), that route traffic to and from the digital subscriber line access multiplexers (DSLAM) and the Internet. They capture full packet payloads without any sampling for over four million TCP flows. They use per-flow features (such as average, minimum, and median packet size in each direction) and the first four packets of individual flows for classification. They also justify $k = 4$, since it is the minimum k that offers good accuracy and precision per application. They consider flows with a 3-way handshake, flows with 3-way handshake plus at least 4 packets, flows with 3-way handshake plus FIN RST flags to indicate the end of data transmission. The statistical classification turns out to be useful to mine the flows left unidentified by DPI tools. However, classifiers may suffer from data overfitting. This challenge prevents a simple strategy such as training on the largest PoP with ground truth and deployment on all other sites. They highlight the need to test new classifiers not only on traces collected on a given site, but also on traces collected on different sites. The latter needs to be done on "homogeneous" traces in terms of the type of traffic and capture time.

Given all the different approaches and methods used for the classification of network traffic, there is a need for a common ground to test and compare all these studies. A network traffic classification benchmark is proposed by Lee *et al.* [61] in order to compare the contributions and/or limitations of traffic classification methods. They propose *NeTraMark* as a benchmark based on six design guidelines,

Category	Application/protocol
Web	HTTP, HTTPS
P2P	FastTrack, eDonkey, BitTorrent, Ares, Gnutella, WinMX, OpenNap, MP2P, SoulSeek, Direct Connect, GoBoogy, Soribada, PeerEnabler, Napster Blubster, FileBEE, FileGuri, FilePia IMESH, ROMNET, HotLine, Waste
FTP	FTP
DNS	DNS
Mail/News	BIFF, SMTP, POP, IMAP, IDENTD, NNTP
Streaming	MMS(WMP), Real, Quicktime, Shoutcast, Vbrick Streaming, Logitech Video IM Backbone Radio, PointCast, ABACast
Network Operation	Netbios, SMB, SNMP, NTP SpamAssasin, GoToMyPc, RIP ICMP, BGP, Bootp, Traceroute
Encryption	SSH, SSL, Kerberos, IPSec, ISAKMP
Games	Quake, HalfLife, Age of Empires, DOOM Battle field Vietnam, WOW, Star Sieze Everquest, Startcraft, Asherons, HALO
Chat	AIM, IRC, MSN Messenger, Yahoo messenger IChat, QNext, MS Netmeet, PGPFone, TALK
Attack	Address scans, Port scans
Unknown	-

Fig. 2. Application Categories [61]

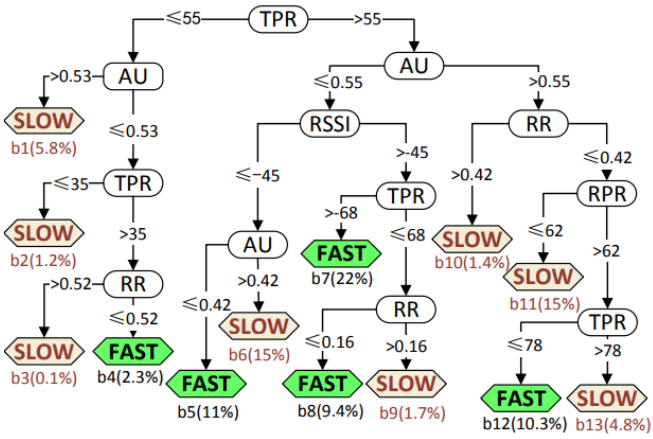

Fig. 3. Pruned decision tree for classifying down link into fast and slow categories. [65]

namely comparability, reproducibility, efficiency, extensibility, synergy, and flexibility/ease-of-use. They consider three categories of performance metrics: flow accuracy (per-whole-trace, and per-application), and *computational performance*. *NeTraMark* incorporates a rich set of eleven state-of-the-art traffic classification methods in its classification engine that are listed in Fig. 2.

Summary: *protocol classification is widely explored by community using supervised methods that are trained on datasets mainly labeled by DPI tools, or unsupervised methods to handle label uncertainty and network parameter changes. The flow definition and features that are used have not converged yet and varies from one study to another.*

B. Network Controlling

Protocol and application identification can be beneficial for improving the Quality of Service (QoS) and Experiment (QoE). In addition to that, there is an increasing demand for efficient network controlling approaches. In this part, we review the applications of ML methods in network controlling, affecting routing policies and congestion control schemes.

Round Trip Time (RTT) estimation is used for various purposes in the network protocols such as determining the proper re-transmission time and congestion. Therefore, the link throughput is mostly affected by the accuracy of RTT estimation. Nunes *et al.* [62] proposed a novel approach for end-to-end RTT estimation using a machine learning technique known as the *fixed-share experts*[63] framework. In their framework, each "expert" guesses a fixed value, the weighted average of these guesses estimates the RTT. Weights are updated after every RTT measurements based on the deviation from the actual RTT. To evaluate the proposed framework, they perform packet level simulation using *QualNet*. Results indicate 40% improvement of estimation accuracy compare to TCP and also Eifel RTT estimator [64].

Pei *et al.* [65] proposes a practical approach toward itemizing the round-trip network latency. They provide the first systematic study on WiFi hop latency in the wild based on the latency and WiFi factors collected from 47 APs on a university campus over two months. Focusing on airtime utilization (AU), queue length (Q), retry ratio (RR), transmitting physical rate (TPR) and receiving physical rate (RPR), they train a decision tree model to understand, troubleshoot, and optimize WiFi hop latency for WiFi APs. For example, they use the trained model to decide whether to move APs closer to the user equipment or not. The model's ruleset is depicted in Fig. 3.

Monteleoni *et al.* [66] apply machine learning methods to tackle the trade-off between energy consumption and performance of wireless devices working on IEEE 802.11. They perform simulation with *ns2* for the evaluation. They define a "loss function" that is based on the wasted performance for sleeping too long or wasted energy for waking up too early or too frequently that is optimized in an online learning manner. Their result suggest that their method is able to save 7 to 20% energy while increasing the latency by a factor of 1.02 to 1.2.

Lutu *et al.* [67] propose the BGP Visibility Toolkit, a system for detecting and analyzing anomalous behavior of BGP routing on the Internet. They show that inter-domain prefix visibility can be used to single out cases of erroneous demeanors resulting from misconfiguration or bogus routing policies. Their dataset consists of a unique ground-truth dataset of 20,000 Low Visible Prefixes (LVPs), that are confirmed by network operators. They use the decision tree (boosted classification trees) method and the *Winnowing algorithm* to predict whether an LVP is intended

or unintended. Their proposed system reveals the patterns of misconfiguration and bogus routing policies which are generally hard to detect.

Cunha *et al.* [68] propose *Sybil* as a system that takes rich queries that researchers and operators express as regular expressions, then returns matched traceroutes even if it has never measured a matching path in the past. Sybil offers a diverse set of traceroute vantage points, using historical measurements to predict which new traces are likely to match a query, task scheduling to optimizes resource usage. They use *RuleFit*, a supervised machine learning technique to assess the confidence in each reported path.

Summary: *Network controlling has different aspects. In this section we consider studies that address challenges in RTT estimation, itemizing RTT delay, BGP routing and visibility checks, and traceroutes. Supervised learning methods and mainly decision tree is a prevalent technique and simulation a frequent approach for validation.*

C. Other Related Networking Topics

There are many studies in the networking domain that are either too specific or too unique to be grouped with other topics. In this section, we review such studies that target specific challenges in network monitoring and controlling.

Network Characteristics: Este *et al.* [69] analyze the stability of the information carried by traffic flow features at the packet level with respect to different capture points (from the edge to the core of the Internet) and capture time (from 2000 to 2008). They illustrate that packet size is stable and the most important feature. Intuitive but was verified that inter-arrival times are half as informative when the traffic is captured in the Internet core as opposed to the Internet edge. They use four classifiers: NN, GMM, K-means, and NB in order to measure the feature importance.

Xu *et al.* [70] address the challenges of modeling the mobile traffic patterns on cellular towers. Their dataset is a month-long network traffic trace logs from an ISP in Shanghai, China. Each entry of the trace contains detailed data usage of 150,000 mobile users, including the ID of devices (anonymized), start and end time of data connection, base station ID, the address of the base station, and the amount of 3G or LTE data usage by each connection. The trace logs 1.96 billion tuples of the described information, contributed by approximately 9,600 base stations all over Shanghai. To model the traffic patterns of large-scale towers deployed in a metropolitan city, they combine three-dimensional information (time, location of towers, and traffic frequency spectrum) and find only five time-domain patterns, indicating the predictability of mobile usage that can be leveraged by providers to serve their customers better.

AS Identification: Measuring, quantifying and understanding the access network performance have been always a challenge due to scale and variation of patterns. Bajpai *et al.* [71] leverage active measurements to investigate patterns and trends in last mile latency by digging into longitudinal datasets from the UK and US. Although they do not use machine learning methods, their approach relies on techniques that can classify ASes by network type, such as effort of Dimitropoulos *et al.* [72] that provide an method to classify ASes using ML techniques back in 2006.

Server OS Identification: Shamsi *et al.* [73] propose *Hershel*, a stochastic model that use SYN packets to classify the deployed OS of a remote server. From the ethical point of view, they have a section explaining why it is not an issue on the security side and can benefit researchers. They leverage the "IRL fingerprinting dataset" with 37.8M samples that contain at least one SYN-ACK packet. Shamsi *et al.* [74] also propose *Heshel+* as a model that can generate signatures in the presence of noise automatically.

Correlated Flows: Zhang *et al.* [75] present *CODA* to automatically identify and schedule Coflows [76] *without* any application modifications. That makes the Coflow more practical and usable by removing the need for manual annotations in applications. CODA employs an incremental clustering algorithm to perform fast, application-transparent Coflow identification, and complements it by proposing an error-tolerant Coflow scheduling to tolerate identification errors. Since Coflows capture a one-off, mutual relationship among some flows that cannot be pre-labeled and need timely identification, they were not able to use traditional classification methods. Therefore, the DBSCAN [77] and incremental version (Rough-DBSCAN [78]) was used.

Frameworks and Enablers: Knowledge plane [79] is one of the first studies on applications of AI in controlling and operating networks. They offer to augment a network with a knowledge plane, a new higher-level artifact that addresses issues of "knowing what is going on" in the network. At an abstract level, this is a system for gathering observations, constraints, and assertions and applying these rules to generate observations and responses. At the physical level, this is a system built out of parts that run on hosts and servers within the network. It is a loosely coupled distributed system of global scope. They argue that cognitive techniques, rather than traditional algorithmic approaches, are best suited to meeting the uncertainties and complexity of their objective. However, that has not been extensively prototyped or deployed in the field yet. In response and in order to prove that such techniques can be used.

Mestres *et al.* [80] discuss a new paradigm called Knowledge-Defined Networking (KDN) to explore the reasons for the lack of adoption and posit that the rise of two recent paradigms: Software-Defined Networking (SDN) and

Network Analytics (NA). They describe a new paradigm that accommodates and exploits SDN, NA, and AI, and provide use-cases that illustrate its applicability and benefits. For the experimental results, they use network trace logs that can be found here [2]. In the learning process, they use the Matlab ANN toolbox with one hidden layer, where the input is the 86 traffic features and the output is the measured CPU consumption.

Bremler-Barr [81] offer *OpenBox*, a software-defined framework for developing, deploying, and managing network-functions. OpenBox decouples the control plane of network-functions from their data plane and allows reuse of data plane elements by multiple logical network functions (NF). In addition to easier management, orchestration, provisioning, and scale, it provides greater flexibility in terms of NF development and deployment, multi-tenancy support with complete tenant isolation, and improved data plane performance. At the core, it uses classifier boxes to make it easy for using ML methods for firewall or intrusion detection and prevention systems (IDPS).

In a visionary paper, Gonzalez *et al.* [82] propose Net2Vec, a modular platform designed to support data scientist in the deployment of machine learning models in the communication network. Net2Vec includes components for capturing, filtering, and manipulation of network data to be analyzed with (deep) machine learning methods. They evaluate the implementation of this platform in another study [83], where they leverage Net2Vet to generate user profiles at line speed from network traces without the need of storing personal information. They use a representation learning approach that is similar to embedding learning approaches in the context of natural language processing. Using two days of HTTP(s) traffic, manually labeled (based on hostnames), they show that Net2Vec is able to accurately profile network users while is more efficient than baseline methods.

Middleboxes: Intermediary network devices (a.k.a middlebox) are widely used in data centers for distributing the load (e.g., load balancers), enabling the remote connectivity (e.g., VPNs), improving the performance (e.g., proxies), and security (e.g., firewalls, IDPS). However, they are responsible for a fraction of failures. Potharaju *et al.* [84] perform a large-scale empirical study of middlebox failures. They have collected network event logs over two years (July 24, 2010-12) across 10+ datacenters hosting 100k+ servers and 2k+ middleboxes; using a wide range of network event sources, including syslog and SNMP alerts, configuration files, a maintenance tracking system, trouble tickets, and traffic data. They mainly focus on these questions:

- How reliable are middleboxes?
- How often do middleboxes fail?
- How long do repairs take?
- How to model failures and repairs?

To answer the last question, they took a formal (mathematical) approach. Using Box-Cox transformation, they transformed data such that it approximately follows the normal distribution. Then using a two-component lognormal mixture, they found the best fit. Then, they focused on Network tickets, analyzed using NetSieve, and focused on main issues and resolution actions. They list several findings that are not aligned with the common views:

- Most failures are due to connectivity errors and link flaps that exhibit intermittent connectivity
- Hardware faults and overload issues are not prevalent
- Misconfiguration (including incorrect rules, VLAN misallocation, and mismatched keys) is a challenge for middleboxes
- Middlebox failover is ineffective in about 33% of the cases for load balancers and firewalls due to the configuration bugs, faulty fail overs, and software version mismatch

Given the large scale challenges, they offer that trend analysis and SDNs are required to reduce the risks and rate of failures due to misconfiguration.

An overview of the ML techniques that are used by each of the studies in this subdomain is provided in Table IV-C.

TABLE II
SUMMARY OF THE ML METHODS USED IN EACH STUDY

ML technique	Studies used
DT	[5], [65], [7], [9], [56], [10], [57]
MLP	[5], [6]
SVM	[7]
BN	[5], [7],[69], [56]
RF	[8], [56], [10]
k-NN	[56],[69]
Clustering	[11], [58],[69]

Section Summary: A wide range of approaches for protocol and application identification and network controlling was reviewed and discussed. It seems that almost in all methods, they focus on solving the issue from a research perspective, without understanding the requirements of implementation in the real world, the type of challenges Internet providers are dealing with, resources they have available, and schemes that are viable. Features selection, training and testing time and computational resource consumption, and approach comparison, given these metrics are still unknown to the research community. We only found a network traffic classification benchmark [61] that propose a benchmark as there is no common ground to compare the contributions and/or limitations

[2]http://knowledgedefinednetworking.org/

of traffic classification methods. Other researchers also found comparison difficult in the absence of a universally agreed metrics and dataset types [85].

V. QoE on the Web and Video Streaming Services

In this section, we focus on the approaches that applied ML techniques on the network traffic in order to improve the Quality of Experience (QoE) measures. Such studies mainly assess the effect of various network measures on the QoE and as a result, improving the rate of user engagement in video streaming, Skype, and web applications.

Improving the user experience while streaming video on mobile platforms has attracted much attention by mobile network providers as a result of a large number of mobile users who are interested in video streaming services. Shafiq *et al.* [12] consider the characterization of mobile video streaming performance to model the impact of performance on user engagement from the perspective of network operators. They suggest that network features exhibit a strong correlation with the abandonment rate and skip rate in video streaming sessions. They assess the correlation using ML techniques to model the relationships between network features and user engagement metrics. They collect 37 million anonymized flow-level logs, from almost half a million users in one month from a tier-1 cellular network in the United States. Their results suggest that decision tree algorithm with bootstrap aggregation (or bagging) is the best choice for both nominal (classification) and continuous (regression) user engagement metrics. They enumerate why they picked DT model: it does not require feature independence assumption and can handle non-linearities by employing multiple splits/breaks for each feature. Furthermore, decision tree models comprise simple if-then-else branches, which can process data efficiently.

Balachandran *et al.* [13] develop a predictive model for Internet video QoE. They discuss two key requirements for the QoE model: 1) it has to be tied into observable user engagement and 2) it should be actionable to guide practical system design decisions. In this study, they systematically highlight challenges in obtaining a robust video QoE model. Then, design a road map for developing Internet video QoE that leverages machine learning. Furthermore, they provide a methodology for identifying and addressing the confounding factors that affect engagement. They also have a practical demonstration of the utility of their QoE model to improve engagement. They pick decision tree as an accurate, intuitive, and actionable model because it can directly be mapped into event processing rules that system designers are typically familiar with them. The data used in this paper was collected on *Conviva.com* in real time using a client-side instrumentation library. This library gets loaded when users watch video on *Conviva.com*'s affiliate content providers' websites. They collect all the quality metrics described earlier as well as play time for each session. In addition, they collect a range of user-specific (e.g., location, device, and connectivity), content (e.g., live vs. Video On Demand, VOD, and popularity), and temporal attributes (e.g., the hour of the day). The dataset that is used for various analysis in this paper is based on 40 million video viewing sessions collected over three months from two popular US based video content providers. The first provider serves mostly VOD content that is between 35 minutes and 60 minutes long. The second provider serves sports events that are broadcast while the event is happening. Their study is limited to clients in the United States. They categorize engagement into different classes based on the fraction of video that the user viewed before quitting. For example, when the number of classes is set to 5, the model tries to predict the portion of video that was watched before quitting in 20% size bins. They begin by using standard machine learning approaches to build a basic predictive model and also to extract the key confounding factors. Having identified the confounding factors, they refine their predictive model to improve accuracy. They consider two different approaches to corroborate confounding features: i) use them as a new feature, ii) split the data and use a new model per feature and combine the model, in which the latter reported to be more accurate.

Sun *et al.* [14] fill the gap between the prediction of the optimal throughput and obtaining the QoE by offering the *CS2P* framework. CS2P leverages data-driven approach to learn i) clusters of similar sessions, ii) an initial throughput predictor, and iii) a Hidden-Markov-Model based midstream predictor modeling the stateful evolution of throughput. One of the main advantages of CS2P is that it can be easily plugged into the bitrate selection logic of client- and server-side adaptation algorithms. They use a proprietary dataset of HTTP throughput measurement from the operational platform of *iQIYI* (an online video content provider). The dataset consists of over 20 million sessions covering 3 million unique client IPs and 18 server IPs over eight days in September 2015. The clients span over 736 cities and 87 ISPs in China. They show CS2P improves QoE by 14% over buffer-based adaptation algorithm.

Jiang *et al.* [15] propose *CFA* (Critical Feature Analytics) as a practical prediction system for video QoE optimization. Although they do not directly use ML methods, similar to the idea of feature selection in ML techniques, they aim to find the best critical features "per" video stream that is dominant, instead of finding the optimal set of features for "all" streams. They develop an algorithm that is based on a domain-specific insight: video quality is typically determined by a subset of critical features, which tend to be persistent. They consider two main properties for prediction models: 1) to be expressive enough to capture complex relationships among video quality and observed session features, and 2) capable of updating quality predictions in near real-time. Their dataset consists of 6.6 million quality measurements collected from 2 million clients using three large public CDNs distributed across 168 countries and 152 ISPs. They benefit

from QoE formulation based on buffer ratio and average bitrate, $QoE = 370BufRatio + AvgBitrate/20$), as defined by Dobrian [86]. They utilize the buffer ratio, average bitrate, join time, and video start failure as quality metrics, and ASN, city, connection type, player, site, Live or VOD, content name , and CDN as features associated with each session. The main insights they share are as follows:

- At a given time, video sessions having the same value on every feature have similar video quality.
- Each video session has a subset of critical features that ultimately determines its video quality.
- Critical features tend to persist on long timescales of tens of minutes.

The importance of user engagement and QoE in the video domain is not limited to downstream video applications. Spetebroot *et al.* [16] offer a new methodology based on machine learning which can link expected QoE to network and device level measurements outside the applications' traffic to predict QoE using *Skype* platform. They use end-to-end delay, packet loss rate, and end-to-end available bandwidth on both up and downstream directions as feature set. The Skype quality meter that is available after 15 seconds helps them to distinguish three different quality classes on 393 different Skype voice calls as follow: 1) Good when the call has no noticeable trouble; 2) Medium when call quality is acceptable but presents minor troubles; 3) Poor when calling quality is terrible with numerous troubles; 4) No call class, that indicates that Skype is not able to establish a voice call because of severe network conditions. Although they mention nine different ML methods that they use for experiments, including DT, ensemble models, and probabilistic models, they only discuss the details of the DT method due to its readability and interpretability.

Toward assessing and improving the Quality of Experience (QoE) on the Web, Da *et al.* [17] aim to thoroughly investigate a mapping between established and recently proposed objective metrics and user QoE. They obtain ground truth QoE via user experiments where they collect and analyze 3,400 Web accesses annotated with QoS metrics and explicit user ratings in a scale of 1 to 5. Their main goal is to narrow the gap between QoS and QoE for Web applications. Their main contributions are 1) method to compute the approximated Above The Fold (ATF) time, 2) dataset collection, and 3) systematically compare expert vs. data-driven models based on a set of QoS metrics. Focusing on the expert models, they compare the results with regression ML methods including Support Vector Regression (SVR), Classification And Regression Tree (CART), and AdaBoost with CART (BOOST) and show the performance of their method is superior compared to these ML methods.

In some cases, ML methods are not the main focus and tool that researchers used. Kelton *et al.* [18] gather label for their dataset by leveraging a ML-enabled eye tracker. They present *WebGaze*, a system that optimizes explicitly for the user-perceived page load time (uPLT) metric. The critical insight in WebGaze is that user attention and interest can be captured using a user's eye gaze and can, in turn, be used to improve uPLT. They collect eye gaze data from 50 users across 45 Web pages and find that there is a commonality in user attention across users. Specifically, users are drawn to specific regions on the page, that they call regions of high collective fixation. WebGaze prioritizes loading objects that exhibit a high degree of collective fixation to improve user-perceived latency.

Section Summary: We reviewed the studies that use the ML techniques as an enabler, a tool, or a baseline to improve the Quality of Experiment (QoS) and as a result user engagement through processing of network measurements. Decision trees and regression methods are the most frequent methods that have been used, mainly due to interpretability or the formulation of the research questions.

VI. Security

The ML techniques have a wide variety of applications in the network security domain. In this section, the advantages of using ML methods in topics such as user profiling, anomaly detection, and security of IoT devices are covered.

A. User Profiling on the Web

User profiling and fingerprinting techniques have been widely studied for a variety of purposes, mainly marketing. Fingerprinting is defined as network level analysis in order to either classify or identify the visited websites based on timing, direction, and volume characteristics of encrypted traffic between a web client and a proxy.

Gonzalez *et al.* [19] examine whether the advent of HTTPS makes profiling harder for anyone beyond the communicating end-points. To this end, they first explain that by knowing the domain that a user visits, either through the server name indication of the TLS protocol or through DNS, an eavesdropper can already derive necessary profiling information, especially for domains whose content is homogeneous. For domains carrying a variety of categories that depend on the particular page that a user visits, e.g., news portals and e-commerce sites, the basic profiling technique fails. Still, accurate profiling remains possible through the transport layer fingerprinting that uses network traffic signatures to infer the exact page that a user is browsing, even under HTTPS. The features they extract from the traffic, when downloading a page, include: the number of incoming packets and the number of outgoing ones, the total size of incoming packets and the total size of outgoing ones, and a trace defined over the size and the order of the observed packets. SVM classifier with an RBF kernel is considered for their experiments. For each monitored website, they capture with TCP-dump the traffic generated by fetching each of the first-level pages 50 times and measure the accuracy of the

classifier using 10-fold cross validation. They consider the model's freshness and its effect on the model's accuracy. Considering daily epochs, they train the model on data from a day and then test it on six consecutive days. As shown in the study, the model degrades faster for dynamic contents over time compare to static websites. They demonstrate that transport fingerprinting remains robust and scalable despite hurdles such as caching and dynamic content for different device types. Overall, their results indicate that although HTTPS makes profiling more difficult, it does not make user profiling unfeasible.

Oh *et al.* [20] incorporate deep neural network to analyze the network traffic and investigate the fingerprinting challenges. They propose three different applications, automated feature engineering, fingerprinting attacks, and fingerprintability prediction. They study the performance of MLP and CNN classifiers on top Alexa websites as well as random websites in binary and multi-class classification, on Onion service fingerprinting, TKS encrypted website fingerprinting, search query fingerprinting, and against four known traffic padding schemes (BuFLO, Tamaraw, WTF-PAD, Walkie-Talkie). In addition, they explain that as a feature extractor, lower dimensional representations, learned by an autoencoder, made recent website fingerprinting attacks more effective and efficient.

As users browse the web, a large list of domains are intentionally (target domains) or unintentionally (domains that host files or provide advertisements) visited. Vassio *et al.* [21] propose a machine learning methodology to extract the services intentionally requested by users, which turn out to be necessary for the profiling purposes. They consider three similarity measures including Jaccard index, maximum likelihood estimation, and cosine similarity based on TF-IDF on the set of domains visited by users to compute their similarity with other users. Among the domains user visits, they call the primary domain that is visited intentionally as *core* and the rest that host the related contents, as *support*. They leverage decision tree to identify the core and support domains. Their features include the length and the content type of the main HTML document (if present); the number of objects of the page and domains contacted by the browser to fetch all objects; HTTP response code (e.g., 2xx, 3xx and 4xx); and whether the browser has been redirected to an external domain. To get the set of features, they perform active crawling and visit the home page of each domain by means of Selenium automatic browser to extract page features. The model's rule set describes the core domains as those with a) the main HTML document size must be bigger than 3357B and b) the browser must not be redirected to an external domain.

Summary: *Multiple ML methods including SVM, MLP, CNN, and DT are applied on user's (encrypted) web browsing data in order to group them with other similar users. Such grouping (user profiling) can be used for marketing and user segmentation purposes.*

B. Anomaly Detection

Deviation from the normal behavior can be a strong signal for detecting attacks in networks. Anomalies can also emerge as a result of routing issues or hardware failure. In all of these case, network providers are interested in detection of anomalies in order to minimize the resulting damage.
Carter *et al.* [22] present a method of detecting anomalous network activities without providing any historical context, using hierarchical clustering. However, it is an O(N^2) operation, meaning that it will not purely scale to large N. This is why they operate their system on a per-service basis, focusing on port 80 in this study. They also used a decision tree to identify the common theme of the detected clusters. To this aim, they labeled data for each cluster as a positive label and all others as negative. DT was able to find the rule mostly based on byte ratio (byte per packet), IP ratio (flow/unique source IP), and outgoing bytes/packet. They experiment and report accurate results on a window of 5 minutes Netflow on port 80 where the server was known to be under SYN flood DDoS attack in that period.

As mentioned by Himura *et al.* [23] the modeling and characterization of Internet traffic are essential for simulations, traffic classification, anomaly detection, and QoS. Focusing on quantifying host-based application, they use a statistical method (multi-scale gamma model) to label anomalies produced in the network in the presence of Internet worms. Using real network traffic traces, they show that applications show consistent behavior over time, however, vary as bandwidth changes. Applications can be characterized using statistical parameters and be distinguished from anomalous behaviors.

Casas and Vanerio [24] leverage the big data analytics and platforms to perform the automatic detection of anomalies in mobile networks. Using Big-DAMA, a big data analytics framework for network monitoring, they apply the Super-learner model [87] (a loss-based ensemble-learning method that finds the optimal combination of a collection of base predictors). They note and tackle two main challenges for the application of ML in anomaly detection: 1) nearly real-time expectation 2) model selection. The first issue is tackled using the Big-DAMA framework, which uses *Spark* for batch (pre-) processing, and *Cassandra* for storage. The benefit of using such a framework is that it is distributed, with no single point of failure, it will be fast and scalable, with the ability to handle unstructured data. To tackle the second issue, they overview the ensemble techniques (bagging, boosting, and stacking) and argue that ensembling can address the issue of selecting the best predictor as we can benefit from multiple predictors. They apply Super learner as a stacking learning algorithm and compare the results with individual ML methods such as decision tree, naive bayes, neural network,

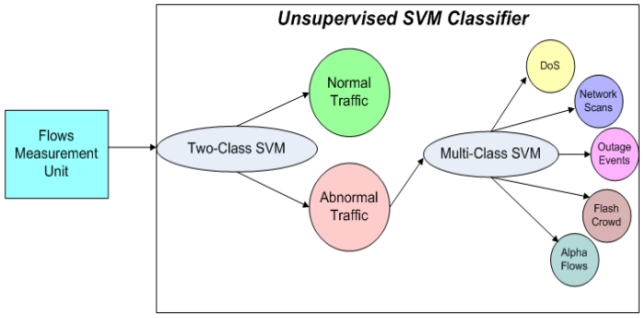

Fig. 4. A high-level representation of the classification framework used by Marnerides [88]

SVM, and k-NN, showing the superiority of ensembling models.

Nevat *et al.* [25] consider the temporally correlated traffic and apply anomaly detection methods. Their dataset is 75-minute TCP traffic of 10th December 2014 from MAWI repository. A statistical method (Markov chain) was used for detection. Their goal is anomaly detection in temporally correlated traffic. They formulate the problem as the optimal statistical test, known as the Likelihood Ratio Test (LRT), using the Cross-Entropy (CE) method. As a result, not only it finds the anomaly but also finds the subset of flows causing it.

In another study, Marnerides *et al.* [88] propose a measurement-based classification framework (illustrated in Fig. 4) that exploits unsupervised learning to categorize network anomalies in specific classes accurately. They introduce the combinatorial use of two-class and multi-class unsupervised Support Vector Machines (SVM) to first distinguish normal from anomalous traffic and to further classify the latter category to individual groups depending on the nature of the anomaly, respectively. The features they have used are as follows:

- Initial flow classification is based on seven packet header characteristics; the IP source/destination addresses, the transport protocol, the transport source/destination ports, the mean packet inter-arrival time and the size of the first ten packets
- During the second stage, only four packet header features are used; the source/destination IP addresses and transport source/destination ports.

Anomaly detection and QoS customization require to know the underlying traffic types. Canini [26] uses flow features to create server profiles and then identifies potential proxies within the observed servers. Their methodology consists of four stages: 1) service identification to identify HTTP and HTTPS services, 2) server profiling - the features are based on the packets' inter-arrival times and payload sizes. A profile consists of the average and standard deviation of this features-, 3) proxy identification using K-means as an unsupervised technique, and 4) host cache management, which is a practical need of storing, updating and deleting service profiles. In this visionary paper, they share preliminary results of experiments on two proxies: `guardster.com` and `anonymouse.org`, where they recorded the traffic to browse a dozen of popular websites using the direct connection and through the proxies, reaching a total of 81 servers. Results show that their method can profile and distinguish among these servers accurately.

Aqil *et al.* [27] consider the network intrusion detection at the scale of a wide area network (WAN) using the ML methods. First, they focus on creating packet summaries that are concise, but sufficient to draw highly accurate inferences. Then, they transform traditional IDS rules to handle summaries instead of raw packets. Using the network traces from MAWI group, they inject five different network attacks such as SYN flooding and extensive port scans at a rate of less than 10% of benign traffic. Showing that Jaal reduces overheads by over 65% compared to sending raw packets while achieving a detection accuracy of over 98% in ISP scale.

Summary: *Various techniques including stacking baseline models, statistical methods, and supervised and unsupervised methods leverage in this section to detect anomaly in the the network traffic. Real time processing and being independence from prior knowledge are among the system design requirements in this domain.*

C. Security and Privacy of IoT

As development and deployment of the Internet of Things (IoT), ranging from a simple body or environmental sensors to home appliances, grow rapidly, the related security and privacy issues become a major concern. As a single penetration to an IoT would lead to a large scale attack. An example would be *Mirai* malware that used IoT devices such as digital cameras to launch destructive DDoS attacks on the Internet. The main concerns regard the devices and protocols, type of attacks, and implications are reviewed in the "Special Issue on Security and Privacy of IoT" [28].

Apthorpe *et al.* [29] analyze the privacy concerns related to traffic analysis of smart IoT devices, even when the content of the traffic is encrypted. They considered a sleep monitor, security camera, smart switch, and Amazon Echo connected to a Raspberry PI as a wireless hotspot. They report that only by analyzing the DNS queries, one can detect what kind of IoT devices are used, although detection specific products with the similar brand will be challenging as their server IP address will be the same. Then, they show that by analyzing the traffic patterns, one can detect when the user is asleep or awake, when is commanding the Amazon Echo, when the switch is turned on or off, or when the camera detected a

motion. However, first, we need to group flows based on their target addresses. This kind of information along with the rest of the analysis that can be done on the users' traffic will provide information about user behaviors and living patterns.

From the same research group, Doshi *et al.* [30] suggest that IoT devices usually communicate to a limited number of endpoints with a rather fixed time intervals between the packets they send. Therefore, they suggest that by focusing on the stateless features (such as packet size, inter-packet interval, and protocol) and stateful features (such as bandwidth and destination IP address cardinality and novelty) DDoS attacks can be accurately be detected in IoT devices using a wide range of ML techniques: K-nearest neighbors, support vector machine with linear kernel (LSVM), decision tree using Gini impurity scores, random forest using Gini impurity scores, and neural network (4-layer fully-connected feedforward neural network) and reported effective accuracy measures (F1, accuracy, precision, and recall) across almost all methods.

Dhakal *et al.* [31] propose a system that is customized to perform online learning of the behavior of residents in a home or business. They use sensor inputs (keypads and camera) to detect whether one of the residents is around or an intruder. They leverage four classifiers on four different data inputs: 1) They use the time difference between the activation of different sensors (time between moving from door to kitchen) to train a K-NN classifier, 2) They apply OpenFace face recognition software to assess the faces from the camera pictures, 3) They consider the number of times the user enters the pin before a successful entry, 4) They focus on the time needed by homeowners to deactivate the alarm and complete the entry event. Finally, the stability and scalability of their method is discussed.

Section Summary: We reviewed studies that leverage ML techniques in the security and privacy domain. Statistical methods as well as decision tree, SVM, and ML-inspired frameworks are leveraged and proposed in order to detect anomalies and perform user profiling with personalization and security implications.

VII. Cloud/Cluster computing

Cloud computing is a response to today's computational-hungry applications with real time and scalability requirements. In this section, we review studies on cluster processing and cloud computing that benefit from utilizing different ML methods.

In distributed systems, clock synchronization is essential to obtain consistency and performance. Geng *et al.* [32] present *HUYGENS*, a software clock synchronization system that uses a synchronization network. They consider CPU and bandwidth usage as well as deployment overhead in their system design. The key ideas are as follows: 1) filtering data received from probes which suffer from queuing delays, random jitter, and Network Interface Card (NIC) timestamp noise, 2) processing the purified data with SVM to accurately estimate one-way propagation times and achieve clock synchronization to within 100 nanoseconds, 3) exploiting a natural network effect - assuming pairwise sync leads to transitively synchronized entities - to detect and correct synchronization errors even further. They use soft-margin SVMs which can tolerate points in the forbidden zone and other noise and delays. Experimenting on two real-world testbeds with different configurations, they have achieved the accuracy to 10s of nanoseconds even at high network load while utilizing less than half a percent of bandwidth and CPU across the network.

Some studies explore the optimization of the whole parallel computing process. In a shared distributed environment, predicting the duration of each job is essential for resource management and job scheduling. *Ernest* was proposed by Venkataraman *et al.* [33] as a performance prediction framework for large scale analytics. They offer a high accuracy and low training overhead using optimal experiment design, which is a statistical technique that allows operators to select the most useful data points for training. At a high level, the user provides a parallel job (written using any existing data processing framework) and a pointer to the input data for the job. Without any dependency on prior knowledge, *Ernest* first determine what training data points to collect, then the features should be derived from the training data, and finally, it performs feature selection to pick the simplest model that best fits the data. They use regression - non-negative least squares (NNLS) model (in SciPy) - leveraging the number of machines used to run the job and the scale (number of rows) of the input data, as well as the overhead of scheduling and communication (as a fraction of number of machines).

Hsieh *et al.* propose *Gaia* [34] a geo-distributed ML system that 1) employs an intelligent communication mechanism over Wide Area Networks (WAN) to efficiently utilize the scarce WAN bandwidth, while retaining the accuracy and correctness guarantees of an ML algorithm; and 2) is generic and flexible enough to run a wide range of ML algorithms, without requiring any changes to the algorithms. They present a new ML synchronization model, *Approximate Synchronous Parallel* (ASP), in which the key idea is to dynamically eliminate insignificant communication between data centers while still guarantee the correctness of ML algorithms. They prototype across 11 Amazon EC2 global regions and a cluster that emulates EC2 WAN bandwidth. *Gaia* can perform ML methods within the 0.94-1.4 of running ML methods on LAN while reducing the cost significantly.

Canali *et al.* [89] analyze the problem of identifying groups of VMs in a cloud infrastructure that exchange information to support a network-aware VMs allocation without having access to a detailed model of the data transfer among them. Due to the popularity and challenges of horizontally replicated vertical stacks, they focus on this scenario. In this

case, each VM in a vertical stack corresponds to a tier of an application, so the scalability is achieved through replication. In order to discover communicating VMs, they leverage the correlation between the time series of network traffic of each VMs and later on clustering algorithms for identifying the vertical stacks of VMs. Their experiments show that the use of ranking-based techniques (as in the Spearman correlation index) outperforms traditional correlation metrics, such as the Pearson index. They compare three clustering algorithms to identify the most suitable alternative for detecting the vertical stack, indicated that Spectral clustering far outperforms the alternatives. Finally, they point out that fine-grained network data collection is essential to achieve high accuracy in the identification of communicating VMs, based on the sensitivity analysis they have performed.

In a related study, Morales *et al.* [90] consider the challenge of over-provisioning in statically managed virtual network topologies (VNT), which are designed to cope with the traffic forecast. Their method (*VENTURE*)'s goal is to reduce expenses (number of required transponders) while ensuring the expected quality of service (maximum bit rate). They propose to reconfigure the VNT based on the predicted traffic, thus adapting the topology to both the current and the predicted traffic volume and direction. To provide a robust and adaptive traffic model, they utilize neural network (ANN) model with an algorithm to set the associated parameters. The reconfiguration problem is modeled mathematically, and a heuristic is proposed to solve it in practical times. their simulation results in SYNERGY test-bed show 8% to 42% improvement in the number of required transponders.

Section Summary: The application of machine learning methods in cloud and cluster computing was reviewed. Studies use a range of ML techniques such as regression (SVR) and clustering (Spectral clustering) to tackle issues such as clock synchronization and predicting the performance of large scale parallel tasks. Some also considered the challenges of applying ML learning in distributed manner and addressed issues such as bandwidth usage in geo-distributed WAN systems.

VIII. Network Traffic Prediction

The time-series are defined as data points with temporal ordering. Such data points are available in a wide range of domains such as weather, bio-medical and bio-metrics, financial (stock and exchange rates), industrial sensors, and also video and music streaming, for which the time-series analysis is applicable. Time-series analysis mainly includes prediction/forecasting, classification, and clustering techniques.

A. Network Traffic Forecasting

Nikravesh *et al.* [35] focus on the importance of resource management for the mobile network providers as the number of subscribers is increasing, and efficiency has become a must-have quality. They apply data analysis techniques to predict the future behavior of mobile network traffic and support network operators to maximize resource usage; preventing both under-provisioning and over-provisioning. They employ a real-life dataset from a commercial trial mobile network composed of a million rows and 27 features, each row representing aggregated (per hour) traffic of one specific cell in the network from 6K different wireless network cells. Although, for their experiments, they only focused on one network cell with the most data points and applied feature selection to exclude non-correlating features, ended up using only 168 data points and 24 features. The problem was formulated as a supervised regression. Their experiment has two folds: 1) to show if the values of attributes can be used to predict the value of a single unknown attribute (as target class), and 2) if prior values of each attribute can be used to predict its next step's value. They compare the accuracy of a fully-connected Neural network in the prediction of future behavior of mobile networks (number of active pieces of equipment in downlink), compare to SVM and Multi-Layer Perceptron with Weight Decay (MLPWD). They show that MLPWD with sliding window is the best choice if the traffic data is uni-dimensional. Otherwise, SVM is a better option.

Nie *et al.* [91] propose a deep belief network and Gaussian model based traffic prediction. The proposed method first adopts discrete wavelet transform to extract the low-pass component of network traffic, describing the long-range dependence of traffic. The low-pass component is modeled by learning a deep belief network. The rest high-pass component that expresses the fluctuations of network traffic, is modeled by a Gaussian model. Maximum likelihood method is used for estimation of the parameters of the Gaussian model. Their dataset is not described in details, but they use the first 2000 data points for training and 16 data points for test. They compare their model with two older studies (principal component analysis (PCA) method [92] and the Tomogravity method [93]), as well as the Sparsity Regularized Matrix Factorization method (SRMF) by Roughan *et al.* [94]. Their proposed prediction method outperforms three existing methods based on spatial and temporal relative errors (SRE, TRE).

Zang *et al.* [36] propose a framework for cellular traffic prediction by pre-processing the time-series using wavelet transformation. First, they apply K-means with the Euclidean distance between base stations as the distance metric, to identify geographically correlated base stations. Then, they apply the wavelet transformation to obtain the low and high-frequency components. They utilize Elman-NN to predict each of these four components and then using them, reconstruct the predicted frequency. Their goal is to predict the next hour traffic value. A real-world traffic data measurement is used to test their framework. Having data from 358 BSs at a particular district in a metropolitan, that each of the BSs records the volumes of GPRS flows over time (hourly). After K-means clustering, the shape of traffic

becomes a 50168 matrix with 144 columns for training and the last 24 columns for prediction and testing. For evaluation, Normalized Mean Square Error (NMSE), Normalized Mean Absolute Error (NMAE), and Mean Absolute Percentage Error (MAPE) was utilized. They discuss the superiority of wavelength transformation over traditional methods.

In another study on cellular traffic prediction, Wang *et al.* [37] capture the spatial-temporal dynamics of cellular traffic by in-cell and inter-cell decomposition using a graph-based deep learning approach. For example, in a transit station, the inter-cell traffic surges at a particular time while in other locations the in-cell traffic can easily dominate. They employ GNN toolkit to implement their experiments and compared the results with LSTM, ARIMA and NAIVE (that predicts the traffic at time t, based on the traffic at time t of the last day). They study the characteristics of urban cellular traffic with large-scale cellular data usage dataset covering 1.5 million users and 5,929 cell towers in a major city of China. Their dataset has the flow-level data per user and covers the app-ID and device-ID for 14 days. They use the first 12 days data (aggregated every half an hour) for training and the last two days for testing. Evaluation is performed using the Mean Absolute Error (MAE) and Mean Absolute Relative Error (MARE). Based on their experiments, they conclude that spatial dependency and the interaction of spatial and temporal factors play an important role in accurate and robust prediction.

Azzouni *et al.* [38] consider the Network Traffic Matrix (TM) prediction to estimate the future network traffic from the previous network traffic data using Long Short-Term Memory (LSTM) models. Considering the linear models such as ARMA (Autoregressive Moving Average), ARIMA (Autoregressive Integrated Moving Average), ARAR (Autoregressive Autoregressive) and HW (HoltWinters) algorithm, they compare the results with nonlinear time series prediction with neural networks. They incorporate a real traffic data provided by GEANT[3] backbone networks, which is the pan-European research network. Using the 15 minutes intervals over three days, they have 309 matrices of network traffic. Using the traffic matrix of N nodes in the network (NxN) over time T (NxNxT), they concatenate rows from top to bottom to create a vector of size (NxN) per time t. A learning window is used to avoid high computational complexity (as the total number of time slots become too big over time). Mean Square Error (MSE) is their evaluation metric for the prediction accuracy. MSE is a scale-dependent metric which quantifies the difference between the forecasted values and the actual values. They discuss the MSE over a different number of hidden layers and hidden units.

Given the wide usage of RNNs for traffic prediction, Vinayakumar *et al.* [39] analyze the variation of Recurrent Neural Networks (RNNs) to obtain the optimum flavor along with optimal parameters. Considering different layer numbers, hidden units, and learning rates, on multiple flavors of RNNs, they propose an architecture of traffic matrix with a sliding window on layers of stacked LSTM. Choosing the parameters to be five layers with 500 units and a learning rate of 0.1. They use the GEANT public dataset for their experiments.

In a study on prediction of TCP throughput, Mizana *et al.* [40] leverage Support Vector Regression (SVR) to predict the TCP throughput based on basic path characteristics including available bandwidth (AB), queuing delays (Q), and packet loss (L). They experiment with passive measurements with multi-configuration testbed. Comparing the accuracy of SVR to the exponentially weighted moving average (EWMA) History-Based Predictor (HB) reveals that for bulk transfers in heavy traffic, TCP throughput is predicted within 10% of the actual value for 87% of the time. Overall, their results suggest approximately a 60% improvement over history-based methods with a much lower impact on end-to-end paths. They use the *relative prediction error* as evaluation metric, which is the predicted throughput (R1) minus actual throughput (R) over minimum of R1 and R. They conclude that while *AB* feature is not necessary for accurate prediction, a combination of queuing delays (Q), and packet loss (L) is sufficient.

Summary: *Network traffic prediction is a complex question to be answered, hence, various techniques (traffic matrix, spatial and temporal separation) and a wide range of models (CNN, LSTM, NN, ARIMA, HW, SVR) applied on the network traffic data in order to predict the future behavior of the network accurately.*

B. Ideas to Improve the Prediction

In addition to traditional studies that address a challenge in the system by forecasting the network traffic and explain how ML methods can lead to more accurate models, there are studies that are inspired by tricks and concepts from other domains and their goal is to improve the forecasting performance. In this subsection, we review such ideas.

Inspired by the idea of Convolutional Neural Networks (CNN) with a sparse neural connection that shows comparable performance compared to the conventional CNNs, Hau *et al.* [41] propose Random Connectivity LSTM (RCLSTM) that contains fewer parameters (35% less neural connections) and show competitive performance. In their model, they initialize a random graph, where neurons connections are established independently in a stochastic manner with a probability of p. The connection is established if $p[i \rightarrow j] \geq T$ where threshold T indicates the sparsity of neural connectivity. Their experiments are based on GEANT backbone network traffic data from 75 workdays, normalized by average over standard deviation. They compare the effect of the number of training samples and length of input traffic (range of 10 to 55) on MSE and MAE accuracy measures, reporting accurate

[3]https://www.geant.org/Projects/GEANT_Project_GN4/

models on larger and longer input traffic.

De-trending is a well-known method to decompose the time series for statistical analysis. Recently, Dai *et al.* [42] propose that de-trending can improve the performance of LSTM models. Therefore, they decompose the traffic into a trend (capturing the fixed temporal pattern) and residual (used for prediction) series. They propose *DeepTrend* as a deep hierarchical neural network. Their model has two layers: a) a fully connected NN called extraction layer and b) an LSTM layer used to make flow prediction. To calculate the *trend*, they calculate the average of the periodic traffic flow time series collected in the same station and subtract it from the actual time-series. Their dataset is collected from 3.9K stations every 5 min in district 4 of freeway systems across California. Using the first 12 weeks for training and four weeks for testing. They consider stations wit less than 1% loss and also normalized the data per station to be 0 mean and 1 standard deviation. Using mean square error (MSE) and mean absolute error (MAE). DeepTrend was able to model the time-series more accurate compared to LSTM models without detrending.

The prediction uncertainty is essential for assessing how much to trust the forecast produced by the model and has a profound impact on anomaly detection. Zhu and Laptev [43] focus on estimating the uncertainty in time-series predictions using neural networks on Uber data. They mention that focusing on Bayesian Neural Networks; the prediction uncertainty can be decomposed into three types: model uncertainty, inherent noise, and the model misspecification. For each, they have proposed a way for calculation for example: For the model uncertainty, given a new input x, they compute the neural network output with stochastic dropouts at each layer. That is, randomly dropout each hidden unit with certain probability p. This stochastic feed-forward is repeated B times. Then, the model uncertainty can be approximated by the sample variance. The model mis-specification deals with the uncertainty when predicting unseen samples with very different patterns from the training dataset. To address that, they use an encoder-decoder. Using the encoder, they extract the representative features from a time series (decoder will, later on, reconstruct the time series from the encoded space). Then, at test time they measure how close is the encoding of the test samples to the training samples. Using the trip data from Uber on eight representative large US cities with three years for training and the following four months for validation and eight months for testing. They log-scales the data, removed the first-day value from the rest (to remove the trend) and fed the data to four models (including a two-layer stacked LSTM with 128 and 32 units and tanh activation). They use the sliding window with a 1-day step size and the Symmetric Mean Absolute Percentage Error (SMAPE) as the performance metric.

Given the proved performance of Convolutional Neural Networks (CNNs) in image classification domain, researchers considered different approaches to convert time series to images and benefit from CNNs. However, there are a small number of studies related to this approach in network domain.

Using the 1D time series signals as the input of a modified CNN architecture, transforming the 1D to 2D matrices and then applying the CNN [44], [45], and also using multiple CNNs and using a fully connected NN to leverage multiple features are among the most popular approaches.

Focusing on road traffic, Ma *et al.* [46] present a method of predicting network-wide traffic based on encoding a day of traffic as an image. This encoding maps time to the x-axis, sampled network locations to the y-axis and represents the traffic at a given (location, time)-point as a single-dimensional "color" value. A convolutional neural network is then trained over previous images and used to predict future network behavior based on past network behavior. A comparison with other machine learning methods is performed using traffic data collected from taxis on two subnetworks of the Beijing road system demonstrating an improved accuracy of up to 43% within acceptable execution time.

Encoding of time-series signals to images using Gramian Angular Fields (GAF) and Markov Transition Fields (MTF) and then using a tiled CNN is used by Wang *et al.* on multiple datasets [44], [45]. Hatami *et al.* [47] propose to use the Recurrence Plots (RP) to transform the time-series into 2D texture images and then take advantage of a deep CNN classifier (with two hidden layers followed by a fully connected layer). Results suggests a boost in performance of Time-series classification rate.

Section Summary: Traffic prediction in mobile networks is necessary as the number of clients is increasing over time and resource management becomes more challenging. We reviewed the applications of ML methods on network traffic prediction as well as ideas on how to improve the accuracy of utilized methods. As mentioned, network traffic prediction is a complex question to be answered, hence, various techniques (traffic matrix, spatial and temporal separation) and a wide range of models (CNN, LSTM, NN, ARIMA, HW, SVR) applied on the network traffic data in order to predict the future behavior of the network accurately. To improve the accuracy, novel ideas from other domains, including randomized connection in LSTM, 2D data structure, and conversion to image and benefiting from CNN models are utilized.

IX. Discussion

There are a large number of studies that leverage ML techniques or statistical approaches to provide monitoring and controlling services for network administrators. However, very small number of such methods are used in the real world scenarios. Liu *et al.* [95] explore the root causes and report that requiring paramter tuning in a manual and iterative way and configuration of thresholds are among the main reasons

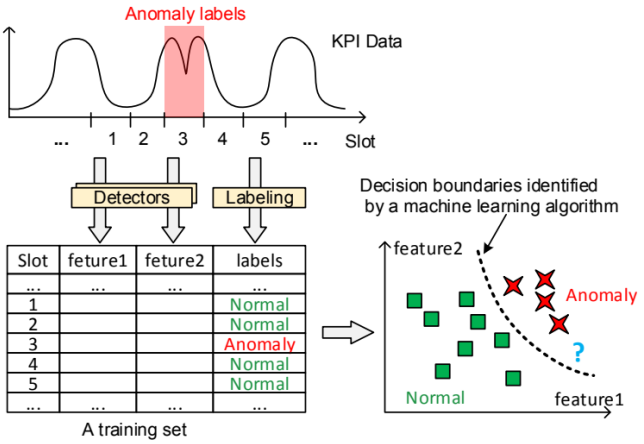

Fig. 5. Opprentice work process [95]

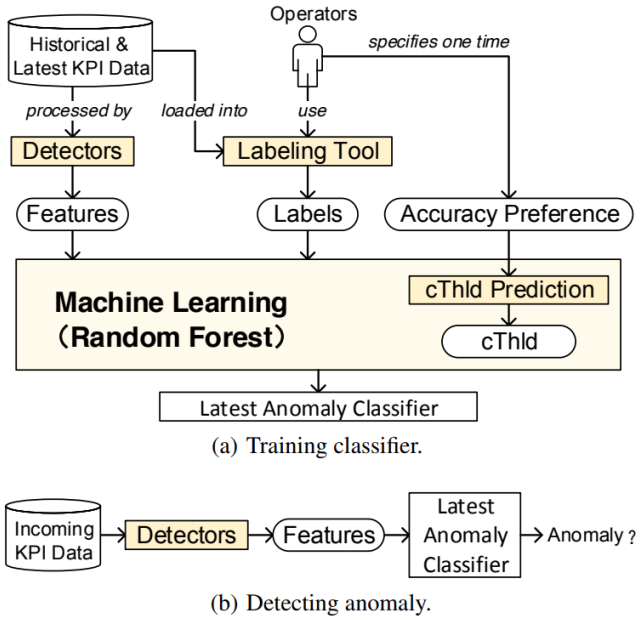

Fig. 6. Opprentice Architecture [95]

that offered monitoring services are not practical. This paper tackles this challenge by *Opprentice* (**Op**erators' ap**prentice**), that operators' only manual work is to periodically label the anomalies in the performance data with a convenient tool. Multiple existing detectors are applied to the performance data in parallel to extract anomaly features. The Opprentice architecture is shown in Figure 6 and its work process is depicted in Figure5. Then the features and the labels are used to train a **random forest classifier** to automatically select the appropriate detector-parameter combinations and the thresholds. With only 2 parameters and robust in existence of noise. For simplicity, no feature selection was done. Recall and precision are used as accuracy criteria and reported to by above 66% for three different service KPIs in a top global search engine, as follows:

- search **page view** (PV)
- **slow responses** of search data centers (#SR), which is an important performance metric of the data centers.
- **search response time (SRT)** This KPI has a measurable impact on the users' search experience

In a detailed study [96] (find slides here[4]) the reasons behind the limited application of ML-based approaches for intrusion detection in large-scale and operational environments are discussed. However, such limitation in some cases can be applied to all of the discussed domains. They explain how the ML approaches are not suitable for the task of intrusion detection:

- lack of training data;
- a very high cost of errors;
- a semantic gap between results and their operational interpretation;
- enormous enormous variability in input data; and (v) fundamental difficulties for conducting the sound evaluation.

They have provided examples of why ML approaches work in some domains and why not for intrusion detection. For example, since the variation of attacks/anomalies are known in recommender systems or spamming, there will be enough training data on both (all) classes, therefore, ML approaches can be used effectively in the real world.

In terms of High cost of errors, they compare with Product recommendation systems, OCR technology, and Spam detection to make the point that in these cases the error of false positives/negatives are not as vital as it is in the intrusion detection domain.

The semantic gap refers to the fact that in the intrusion detection domain, operational expectations are more than what a current ML technique can deliver. Instead of a large number of false positives that only mark unseen behaviors (that can be legitimate or attack), it is expected that the results be interpreted for the operators.

The local and internal policies should also be considered in addressing the semantic gap. Addressing such policies as well as vague guidelines described by an imprecise legal language are the main issues.

In terms of Diversity of Network Traffic, as network characteristics are variable over short time ranges (seconds to hours), it will be hard to predict them. Aggregating network traffic (time of the day, the day of the week) can be a better solution and work better.

Summary: *All in all, very small number of studies proposed a practical approach for utilizing the benefits of ML in networking system, each either focused on a very specific task or do not discuss and address the limitations of real-world scenario deployment.*

[4]Slides: HTTP://oakland10.cs.virginia.edu/slides/anomaly-Oakland.pdf

X. Conclusion

This survey reviewed a collection of networking studies that employ ML techniques. We summarized them into groups based on their context and then primarily examine how they use ML techniques and described related issues. Our examination has led to a number of observations in how these methods are used by the networking community in recent years as follows:

- Hyper-parameter tuning and generalization are not discussed. Therefore, reproducibility will be an issue.
- Feature selection either based on feature importance analysis or sequential forward selection techniques is not considered in most of the studies.
- Real-time and link-rate analysis are one of the most critical system design requirements. ML models are expected to be interpretable and insights to be actionable.
- Data enrichment approach using external sources and aggregated available features are considered as a way to improve the quality of data.
- Performance and security are the most frequently observed areas that these methods have been deployed.
- Recent studies have a tendency toward 1) using real-world data for evaluation, and 2) Deep learning methods, and 3) and end to end system design that covers the whole pipeline, from data collection to deployment.
- Supervised learning (2-classes classification) is more prevalent compare to unsupervised methods.
- Quality of labeled data is an issue because it is difficult to obtain labels in a reliable and scalable manner. The most common approach is DPI where packet payload is used to label flows. However, this approach often has limited coverage as the content is not available or is encrypted.

References

[1] S. Hao and H. Wang, "Exploring domain name based features on the effectiveness of dns caching," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 1, pp. 36–42, 2017.

[2] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, and T. Holz, "Going wild: Large-scale classification of open dns resolvers," in *Proceedings of the Conference on Internet Measurement Conference*, IMC, pp. 355–368, 2015.

[3] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis.," in *Ndss*, 2011.

[4] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the Conference on Internet Measurement Conference*, IMC, pp. 48–61, 2010.

[5] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[6] A. K. J. Michael, E. Valla, N. S. Neggatu, and A. W. Moore, "Network traffic classification via neural networks," tech. rep., University of Cambridge, Computer Laboratory, 2017.

[7] X. Yun, Y. Wang, Y. Zhang, and Y. Zhou, "A semantics-aware approach to the automated network protocol identification," *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 1, pp. 583–595, 2016.

[8] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking (ToN)*, vol. 23, no. 4, pp. 1257–1270, 2015.

[9] P. M. Santiago del Rio, D. Rossi, F. Gringoli, L. Nava, L. Salgarelli, and J. Aracil, "Wire-speed statistical classification of network traffic on commodity hardware," in *Proceedings of the Internet Measurement Conference*, IMC, pp. 65–72, 2012.

[10] Y. Wang and R. Nelson, "Identifying network application layer protocol with machine learning," *PAM-International Conference on Passive and Active Network Measurement*, 2009.

[11] G. Szabó, J. Szüle, Z. Turányi, and G. Pongrácz, "Multi-level machine learning traffic classification system," in *Proc. of the 11th IEEE Intl Conf. on Networks (ICN)*, pp. 69–76, 2012.

[12] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Understanding the impact of network dynamics on mobile video user engagement," in *The ACM International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS, pp. 367–379, 2014.

[13] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 339–350, 2013.

[14] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the ACM SIGCOMM Conference*, pp. 272–285, 2016.

[15] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "Cfa: A practical prediction system for video qoe optimization.," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 137–150, 2016.

[16] T. Spetebroot, S. Afra, N. Aguilera, D. Saucez, and C. Barakat, "From network-level measurements to expected quality of experience: the skype use case," in *Measurements & Networking (M&N), IEEE International Workshop on*, pp. 1–6, 2015.

[17] D. Da Hora, A. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between qos metrics and web qoe using above-the-fold metrics," in *PAM-International Conference on Passive and Active Network Measurement*, pp. 1–13, 2018.

[18] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving user perceived page load times using gaze.," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 545–559, 2017.

[19] R. Gonzalez, C. Soriente, and N. Laoutaris, "User profiling in the time of https," in *Proceedings of the Conference on Internet Measurement Conference*, IMC, pp. 373–379, 2016.

[20] S. E. Oh, S. Sunkam, and N. Hopper, "Traffic analysis with deep learning," *arXiv preprint arXiv:1711.03656*, 2017.

[21] L. Vassio, D. Giordano, M. Trevisan, M. Mellia, and A. P. C. da Silva, "Users' fingerprinting techniques from tcp traffic," in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pp. 49–54, 2017.

[22] K. M. Carter, R. P. Lippmann, and S. W. Boyer, "Temporally oblivious anomaly detection on large networks using functional peers," in *Proceedings of the Conference on Internet Measurement Conference*, IMC, pp. 465–471, 2010.

[23] Y. Himura, K. Fukuda, K. Cho, and H. Esaki, "Quantifying host-based application traffic with multi-scale gamma model," in *PAM-International Conference on Passive and Active Network Measurement*, pp. 2–3, 2009.

[24] P. Casas and J. Vanerio, "Super learning for anomaly detection in cellular networks," in *Wireless and Mobile Computing, Networking and Communications (WiMob),*, pp. 1–8, 2017.

[25] I. Nevat, D. M. Divakaran, S. G. Nagarajan, P. Zhang, L. Su, L. L. Ko, and V. L. L. Thing, "Anomaly detection and attribution in networks with temporally correlated traffic," *IEEE/ACM Transactions on Networking (ToN)*, vol. 26, no. 1, pp. 131–144, 2018.

[26] M. Canini, W. Li, A. W. Moore, *et al.*, "Toward the identification of anonymous web proxies," in *PAM-International Conference on Passive and Active Network Measurement*, 2009.

[27] A. Aqil, K. Khalil, A. O. Atya, E. E. Papalexakis, S. V. Krishnamurthy, T. Jaeger, K. Ramakrishnan, P. Yu, and A. Swami, "Jaal: Towards network intrusion detection at isp scale," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pp. 134–146, 2017.

[28] H. Wang, Z. Zhang, and T. Taleb, "Special issue on security and privacy of iot," *World Wide Web*, pp. 1–6, 2017.

[29] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[30] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," *arXiv preprint arXiv:1804.04159*, 2018.

[31] A. Dhakal and K. Ramakrishnan, "Machine learning at the network edge for automated home intrusion monitoring," in *Network Protocols (ICNP), IEEE 25th International Conference on*, pp. 1–6, 2017.

[32] Y. Geng, S. Liu, Z. Yin, A. Naik, B. Prabhakar, M. Rosenblum, and A. Vahdat, "Exploiting a natural network effect for scalable, fine-grained clock synchronization," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, USENIX Association, 2018.

[33] S. Venkataraman, Z. Yang, M. J. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics.," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 363–378, 2016.

[34] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds.," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 629–647, 2017.

[35] A. Y. Nikravesh, S. A. Ajila, C.-H. Lung, and W. Ding, "Mobile network traffic prediction using mlp, mlpwd, and svm," in *Big Data (BigData Congress), IEEE International Congress on*, pp. 402–409, 2016.

[36] Y. Zang, F. Ni, Z. Feng, S. Cui, and Z. Ding, "Wavelet transform processing for cellular traffic prediction in machine learning networks," in *Signal and Information Processing (ChinaSIP), IEEE China Summit and International Conference on*, pp. 458–462, 2015.

[37] X. Wang, Z. Zhou, Z. Yang, Y. Liu, and C. Peng, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," in *IEEE 25th International Conference on Network Protocols (ICNP)*, pp. 1–10, 2017.

[38] A. Azzouni and G. Pujolle, "A long short-term memory recurrent neural network framework for network traffic matrix prediction," *arXiv preprint arXiv:1705.05690*, 2017.

[39] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *Advances in Computing, Communications and Informatics (ICACCI), International Conference on*, pp. 2353–2358, 2017.

[40] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," *IEEE/ACM Transactions on Networking (ToN)*, vol. 18, no. 4, pp. 1026–1039, 2010.

[41] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Traffic prediction based on random connectivity in deep learning with long short-term memory," *arXiv preprint arXiv:1711.02833*, 2017.

[42] X. Dai, R. Fu, Y. Lin, L. Li, and F.-Y. Wang, "Deeptrend: A deep hierarchical neural network for traffic flow prediction," *arXiv preprint arXiv:1707.03213*, 2017.

[43] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *Data Mining Workshops (ICDMW), IEEE International Conference on*, pp. 103–110, 2017.

[44] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," *arXiv preprint arXiv:1506.00327*, 2015.

[45] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, vol. 1, 2015.

[46] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[47] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," in *Tenth International Conference on Machine Vision (ICMV)*, vol. 10696, p. 106960Y, International Society for Optics and Photonics, 2018.

[48] M. Zhang, W. John, K. Claffy, and N. Brownlee, "State of the art in traffic classification: A research review," in *PAM-Student workshop on Passive and Active Network Measurement*, pp. 3–4, Citeseer, 2009.

[49] J. Ashraf and S. Latif, "Handling intrusion and ddos attacks in software defined networks using machine learning techniques," in *Software Engineering Conference (NSEC), National*, pp. 55–60, 2014.

[50] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, 2018.

[51] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[52] K. Fukuda and J. Heidemann, "Detecting malicious activity with dns backscatter," in *Proceedings of the Internet Measurement Conference*, IMC, pp. 197–210, 2015.

[53] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *International Workshop on Passive and Active Network Measurement*, pp. 41–54, Springer, 2005.

[54] M. Canini, W. Li, A. W. Moore, and R. Bolla, "Gtvs: Boosting the collection of application traffic ground truth," in *International Workshop on Traffic Monitoring and Analysis*, pp. 54–63, Springer, 2009.

[55] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Transactions on Parallel and Distributed systems*, vol. 24, no. 1, pp. 104–117, 2013.

[56] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in *Computing, Networking and Communications (ICNC), 2017 International Conference on*, pp. 843–848, 2017.

[57] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in sdn," in *Proceedings of the ACM SIGCOMM Conference*, pp. 487–488, 2013.

[58] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *The IEEE Conference on Local Computer Networks*, pp. 250–257, 2005.

[59] E. Glatz and X. Dimitropoulos, "Classifying internet one-way traffic," in *Proceedings of the Internet Measurement Conference*, IMC, pp. 37–50, 2012.

[60] M. Pietrzyk, J.-L. Costeux, G. Urvoy-Keller, and T. En-Najjary, "Challenging statistical classification for operational usage: The adsl case," in *Proceedings of the Conference on Internet Measurement Conference*, IMC, pp. 122–135, 2009.

[61] S. Lee, H. Kim, D. Barman, S. Lee, C.-k. Kim, T. Kwon, and Y. Choi, "Netramark: A network traffic classification benchmark," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 22–30, 2011.

[62] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning framework for tcp round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, p. 47, 2014.

[63] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine learning*, vol. 32, no. 2, pp. 151–178, 1998.

[64] R. Ludwig and K. Sklower, "The eifel retransmission timer," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 3, pp. 17–27, 2000.

[65] 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 *INFOCOM -The 35th Annual IEEE International Conference on Computer Communications, IEEE*, pp. 1–9, 2016.

[66] C. Monteleoni, H. Balakrishnan, N. Feamster, and T. Jaakkola, "Managing the 802.11 energy/performance tradeoff with machine learning," *Networks and Mobile Systems*, 2004.

[67] A. Lutu, M. Bagnulo, C. Pelsser, O. Maennel, and J. Cid-Sueiro, "The bgp visibility toolkit: Detecting anomalous internet routing behavior," *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 2, pp. 1237–1250, 2016.

[68] Í. Cunha, P. Marchetta, M. Calder, Y.-C. Chiu, B. Schlinker, B. V. Machado, A. Pescapè, V. Giotsas, H. V. Madhyastha, and E. Katz-Bassett, "Sibyl: A practical internet route oracle.," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 325–344, 2016.

[69] A. Este, F. Gringoli, and L. Salgarelli, "On the stability of the information carried by traffic flow features at the packet level," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 3, pp. 13–18, 2009.

[70] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," *IEEE/ACM Transactions on Networking (ToN)*, vol. 25, no. 2, pp. 1147–1161, 2017.

[71] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, "Dissecting last-mile latency characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 5, pp. 25–34, 2017.

[72] X. Dimitropoulos, D. Krioukov, G. Riley, *et al.*, "Revealing the autonomous system taxonomy: The machine learning approach," *arXiv preprint cs/0604015*, 2006.

[73] Z. Shamsi, A. Nandwani, D. Leonard, D. Loguinov, Z. Shamsi, A. Nandwani, D. Leonard, and D. Loguinov, "Hershel: Single-packet os finger-

printing," *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 4, pp. 2196–2209, 2016.

[74] Z. Shamsi and D. Loguinov, "Unsupervised clustering under temporal feature volatility in network stack fingerprinting," in *The ACM International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS, pp. 127–138, 2016.

[75] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "Coda: Toward automatically identifying and scheduling coflows in the dark," in *Proceedings of the ACM SIGCOMM Conference*, pp. 160–173, 2016.

[76] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pp. 31–36, 2012.

[77] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[78] P. Viswanath and V. S. Babu, "Rough-dbscan: A fast hybrid density based clustering method for large data sets," *Pattern Recognition Letters*, vol. 30, no. 16, pp. 1477–1488, 2009.

[79] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 3–10, 2003.

[80] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Ma'ruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, 2017.

[81] A. Bremler-Barr, Y. Harchol, and D. Hay, "Openbox: A software-defined framework for developing, deploying, and managing network functions," in *Proceedings of the ACM SIGCOMM Conference*, pp. 511–524, 2016.

[82] R. Gonzalez, F. Manco, A. Garcia-Duran, J. Mendes, F. Huici, S. Niccolini, and M. Niepert, "Net2vec: Deep learning for the network," in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pp. 13–18, 2017.

[83] R. Gonzalez, A. Garcia-Duran, F. Manco, M. Niepert, and P. Vallina, "Network data monetization using net2vec," in *Proceedings of the SIGCOMM Posters and Demos*, pp. 37–39, 2017.

[84] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: a field study of middlebox failures in datacenters," in *Proceedings of the Conference on Internet Measurement Conference*, IMC, pp. 9–22, 2013.

[93] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," in *SIGMETRICS Performance Evaluation Review*, vol. 31, pp. 206–217, ACM, 2003.

[85] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[86] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 362–373, 2011.

[87] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Statistical applications in genetics and molecular biology*, vol. 6, no. 1, 2007.

[88] A. K. Marnerides, D. P. Pezaros, H.-c. Kim, and D. Hutchison, "Unsupervised two-class and multi-class support vector machines for abnormal traffic characterization.," in *PAM-Student workshop on Passive and Active Network Measurement*, 2009.

[89] C. Canali and R. Lancellotti, "Identifying communication patterns between virtual machines in software-defined data centers," *SIGMETRICS Performance Evaluation Review*, vol. 44, no. 4, pp. 49–56, 2017.

[90] F. Morales, M. Ruiz, L. Gifre, L. M. Contreras, V. López, and L. Velasco, "Virtual network topology adaptability based on data analytics for traffic prediction," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 1, pp. A35–A45, 2017.

[91] L. Nie, D. Jiang, S. Yu, and H. Song, "Network traffic prediction based on deep belief network in wireless mesh backbone networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–5, 2017.

[92] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic matrices: balancing measurements, inference and modeling," in *SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 362–373, ACM, 2005.

[94] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," *IEEE/ACM Transactions on Networking (ToN)*, vol. 20, no. 3, pp. 662–676, 2012.

[95] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the Internet Measurement Conference*, IMC, pp. 211–224, 2015.

[96] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Security and Privacy (SP), IEEE Symposium on*, pp. 305–316, 2010.