Extending Networking Curriculum with Applied Artificial Intelligence

Petr Matoušek
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
matousp@fit.vutbr.cz

Ondřej Ryšavý
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
rysavy@fit.vutbr.cz

Ivana Burgetová
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
burgetova@fit.vutbr.cz

***Abstract*—Artificial Intelligence (AI) and related technologies like Data Mining, Machine Learning or Neural Networks became very popular in recent years. Many IT companies today require graduated students to understand and be able to apply these technologies. Application potential of AI is not limited only to robotics, image processing or intelligent agents, but also in engineering areas like computer networking and communication. However, on most universities, networking courses focus mainly on transmission protocols, network services and hardware design only while AI, machine learning or neural networks are taught separately. This causes a gap that emerges between AI theory and engineering approach. Thus, teachers of engineering courses are challenged how to introduce their students to an application of AI in the engineering areas, e.g., electronics, communication, embedded systems, power grids, etc. This paper shows how selected AI techniques presently used in computer networks can be incorporated into networking curriculum and demonstrated to students which extends student competencies and prepares them better into future jobs. We also present two case studies where AI techniques are applied on networking data in order to solve typical engineering problems.**

***Index Terms*—Computer Networking, Curriculum, Artificial Intelligence, Engineering Education**

I. Introduction

Recent advances in robotics, Machine Learning and Artificial Intelligence (AI) together with requirements on processing massive amounts of data make a new challenge for educational institutes to integrate these technologies into curricula and teach students how to apply AI methods in engineering practice. This is a big imperative especially for Europe where progress in AI and automation is behind the US and China [1]. The call for modernizing education curricula is also mentioned in the Recommendations on AI Policy [2] presented on Tallinn Digital Summit in 2018.

Progress in computer performance and data storages in recent years have enabled machine learning, neural networks and AI to receive extraordinary commercial and research interest. AI methods and machine learning approaches are used for real-time language processing, image analysis, autonomous vehicle control, automated customer service, process control and other domains. As stated by EU Joint Research Center *it is reasonable to expect that the recent advances in AI and machine learning will have profound impacts on future labour markets, competence requirements, as well as in earning and teaching practices* [3].

In order to stay competitive and prepare graduate students for future technologies, educational institutions are challenged to incorporate AI into engineering curricula.

A. Adding AI into Networking Courses

Building a curriculum for graduate engineering programs is not easy today because there is a vast number of emerging technologies and interdisciplinary subjects that are demanded by employers to be covered by education. On the other hand, students still need to be taught principles and fundamentals of traditional technologies. Thus, teachers struggle with the question what techniques and principles should be kept in the curriculum and what should be replaced with new promising technologies while the volume of learning objectives and student workload should remain unchanged.

One solution is to implement new courses focused on recent technologies like Artificial Intelligence, Deep Learning, or Big Data Analysis while keeping traditional engineering courses untouched. This approach, however, increases gap between theoretical and engineering approach and students will still struggle how to apply AI techniques on engineering problems.

Another approach is to incorporate selected AI techniques and methods directly into engineering courses without deep explanation of background theory. On selected case studies students will learn how AI is applied on typical real-world problems and understand relevance of AI for engineering. The case study can be organized as a step-by-step tutorial where input data and scripts implementing the AI techniques will be provided. Thus, a student may follow the instructions and process, analyze and observe given data by himself or herself. The tutorial may include additional assignments which challenge a student to undertake additional experiments.

As a source of cases studies, educators my employ freely-available demonstration, text book resources, or results of own ongoing research where AI is applied.

B. Structure of the text

The paper is structured as follows. Section II gives a short overview of current engineering curricula and discusses their

978-1-7281-3222-8/19/$31.00

priorities. Section III introduces AI methods that are applied in networking. Section IV presents two case studies where AI techniques are applied on networking data. The first case study presents Anomaly Detection of Internet of Things (IoT) communication, the second case study demonstrates Data Mining methods on Domain Name System (DNS) data. Each case study includes a problem description, input data, and applied AI techniques. The last section concludes the paper.

C. Contribution

The main contribution of the paper is to present AI techniques that can be incorporated into networking courses so that students will see relevance of AI in networking and be able to apply selected AI techniques into real-world scenarios. The second contribution includes two case studies which demonstrate interconnection of AI and networking. The studies demonstrate how to extend networking curricula with AI.

II. Engineering Curriculum and AI

By studying recommendations for engineering curricula, e.g., ACM/IEEE-CS Curriculum for Computer Engineering (2016) [4], ACM/AIS Model for Graduate Degree Programs in Information Systems (2016) [5], ACM/IEEE-CS/AIS SIGSEC/IFIP WG 11.8 Guidelines for Post-Secondary Degree Programs in Cyber security (2017) [6], ACM SIGCOMM Curriculum Designs (2002) [7], or Electrical Engineering Curriculum (2017) [8] we may notice two interesting things.

First, it seems that there have not been any attempt to update recommendations for engineering curricula within past three years. The recent changes happened during 2016 and 2017. Since broader attention to AI can be traced in 2017 and 2018, it is not surprising that AI approaches are not a part of the above mentioned recommendations. This creates a challenge for future ACM and IEEE-CS curricula boards to adopt AI techniques into new recommendations for both undergraduate and graduate degree programs.

The second issue that can be observed by studying above mentioned documents is that curricula in electrical engineering still put main emphases on electronics, circuit design, signal processing, i.e., mostly physics-related technologies. These curricula, however, omit data processing and analytics which becomes more and more important application domain for engineers due to huge amount of data that is produced by various physical systems. Traditionally, big data analysis was a domain of data science experts or database specialists. Today, demand on analyzing big data moves to engineering positions.

What is surprising that neither Cyber Security Curriculum [6] published in December 2016 did not include Data Mining nor Machine Learning to process data from security incidents.

All these facts build a new challenge for contemporary educators in engineering courses who should actively search and incorporate selected AI techniques into engineering courses in order to prepare students for their future work and enhance level of their competencies.

The following text discusses what AI techniques are relevant for computer networks and can be incorporated into networking courses.

III. AI methods in Networking

Application of Artificial Intelligence in networking is not a new thing. Machine Learning (ML) techniques have been adopted in many engineering areas, including computer networking domain hoping to provide an efficient solution to more complicated problems in network design and management [9]. Traditionally, Machine Learning have been used for traffic classification [10]. Efficient traffic classification is necessary for intrusion detection and performance prediction. In addition to classical applications, Machine Learning can be used to solve problems of parameters adaptation, network cognitive management, and configuration extrapolation. Automated network resource scheduling and decision making is also a big opportunity for application of ML methods [11].

Various ML techniques have been also exploited for solving problems in network applications. For network traffic classification, supervised ML techniques, such as Naïve Bayes, Support Vector Machine (SVM), Decision Trees and Neural Networks, were used to label network traffic based on previously known applications [12]. Application of supervised techniques requires to prepare training data. This means to label a network flow either manually or by means of another method. Unsupervised methods, such as clustering, were also considered to provide automatic network classification [13] based on significant features, e.g, inter-packet arrival time, packet size, traffic patterns, etc.

Network security is also an important aspect of networking and many ML-based approaches have been proposed for this domain. Commonly, these approaches aim at identifying anomalous traffic which may represent a potential security risk [14]. In addition to traditionally applied techniques, Deep Learning seems to be a promising method for increasing accuracy of network threat detection, in particular, zero-day attacks or Advanced Persistent Threats (APT). Deep Neural Networks (DNN)-based intrusion detection was proposed in [15]. The approach was tested on NSL-KDD dataset that contained ten different types of attacks [16]. The results showed that the DNNs-based Intrusion Detection Systems are reliable and efficient enough to detect specific attack classes.

Machine Learning work-flow comprises of several stages [9]: (i) problem formulation, (ii) data collection, (iii) data analysis, (iv) model construction, (v) model validation, and (vi) deployment and inference. In networking domain, data collection is more complicated due to volatility of information. The network traffic traces need to be collected in real time and at different network spots. While analysis and model training can be done off-line, deployment of the model to the system requires the trade-off between accuracy and performance because real-time input is processed with limited resources.

IV. Case Study 1: Anomaly Detection of CoAP communication Using Profiles

A. Problem Description

Smart home networks with Internet of Things (IoT) devices can be a target of cyber attacks against IoT communication

since many IoT protocols are implemented without authentication or encryption. Attackers can intercept IoT communication and also manipulate with IoT devices by sending spoofed commands to IoT sensors or controllers. This case study aims to demonstrate students the security issues of IoT environment and to show how anomaly behavior can be detected by application of AI methods. The simple testbed consisting Constrained Application Protocol (CoAP) nodes (see Figure 1) is employed to obtain samples of data communication that are provided to students for analysis.

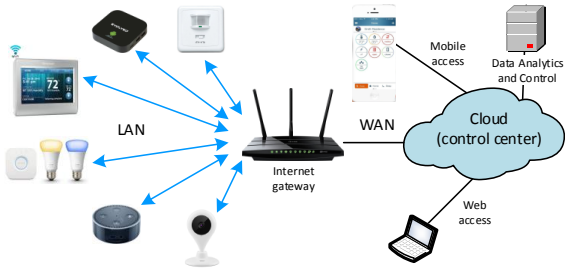

Fig. 1. Topology of IoT ecosystem.

B. Input Data

In this case study, students are provided CoAP traffic [17] which represents monitoring and control data transmission between a client and a server in the test-bed network. The CoAP server is implemented on an IoT device (sensor, actuator, etc.) where it measures physical quantity like temperature and humidity or where it controls motion, light, smoke, etc. For our purposes, we will monitor CoAP communication and extract L7 (application layer) data like a code of the command, token, message ID, and URI. This dataset serves as an input for anomaly-based analysis of security incidents. Apart from regular communication datasets, students are also given samples of selected networks attacks, e.g., an unauthorized resource access, denial of service, etc. The dataset used in this case study consists of a collection of capture files as listed in Table I. Files `idle.cap`, `regular.cap` and `observe.cap` contain normal communication that will be used for learning the communication profile. File `attack.cap` contains samples of anomaly communication for testing the anomaly detection method.

TABLE I
DATASETS USED IN A CASE STUDY

File	Packets	Flows	Resources	Normal
idle.cap	25 096	716	5	yes
regular.cap	54 634	1 307	12	yes
observe.cap	17 480	415	8	yes
attack.cap	38 474	870	8	no

C. Method

As an example of the possible solution to the problem of detecting anomalies in a CoAP network, we provided a simple anomaly detection method based on the statistical model of the communication profile combining approaches by [18] and [19]. The method has two stages. In the learning mode, the method uses input data to learn the normal profile for the system. In the discrimination mode, the learned method is applied to unknown data in order to classify the communication. An overview of the method is as follows:

- The learner observes a typical CoAP communication and builds profiles of communicated devices. The profile corresponds to a statistical *resource usage model* $\mathcal{M}$ that relates to an operation r^{op} on the resource r^{uri}. The statistical information of model $\mathcal{M}$ is characterized by random variables X_1, X_2 that represent the number of packets and the number of octets associated with the resource operation. These variables are used to define a joint probability function $f(x_1, x_2) = P(X_1 = x_1, X_2 = x_2)$. The model characterizes usage of a monitored resource within the specific period of observation which is given by a fixed-size time window.
- Traffic classifier computes the similarity of currently observed behavior to the known behavior represented by corresponding resource usage model $\mathcal{M}$. In the operational phase, the network communication is analyzed as follows:
  1) The traffic is captured within a given time window.
  2) For each flow e resource usage label r is determined.
  3) The expected resource usage model $\mathcal{M} = (f_{X_1,X_2}(x_1, x_2), t)$ is retrieved from the usage profile $\mathcal{P}$.
  4) Flow features are extracted from flow e to form a vector of observations $\vec{y}$.
  5) Finally, the joint probability function of model $\mathcal{M}$ is used to compute the probability for the observed behavior, $p = f_{X_1,X_2}(\vec{y})$. If the probability is greater than threshold t, the flow is marked as *normal*. Otherwise, it is labeled as *abnormal*.

D. Evaluation

An inevitable requirement of this case study is to provide evaluation of the implemented method. This demonstrates students how validation of results is important. Also, it gives them the baseline that can be used for comparison of their solutions. The evaluation is performed by computing *hit ratio* (recall) and *false positives*. For each executed test, the following quantities are measured:

- N: the number of all normal flows in the testing dataset
- N_+: the number of flows correctly classified as normal (true positives)
- N_-: the number of flows incorrectly classified as normal (false positives)

Hit ratio is computed as follows [23]:

$$H_r = \frac{N_+}{N}. \tag{1}$$

False positive is given by the following equation:

$$F_p = \frac{N_-}{N_- + N_+}. \tag{2}$$

Table II presents results of flow classification using different profiles. The profile computed from `idle` dataset has poor hit ratio. It is because that dataset does not contain enough patterns to represent behavior of the system adequately. We also tested profiles based on samples drawn from multiple data files achieving hit ratio around 90%.

TABLE II
PERFORMANCE MEASURES OF BASIC PROFILES

Profile	H_r	F_p
idle	39.91%	2.31 %
regular	75.98%	7.23 %
observe	84.82%	8.17 %
idle+regular	88.72%	6.36 %
idle+regular+observe	90.85%	6.46 %

Providing all these different results together with a proper explanation is invaluable for understanding the issues related to the design of anomaly detection methods. Students will understand that not only properly designed method, but also the choice of a set of learning samples is important to achieve good results.

E. Recommendation

The presented case study aims at demonstrating the application of anomaly detection methods on an IoT network. Students are given datasets consisting of captured communication as the input. While the idea of profile-based anomaly detection is quite simple, its realization can be tricky. Students need to understand the input data, extract relevant features in order to build the model, design a learner and a classifier. A provided guide though relatively simple is supposed to help students in each of the steps towards the solution.

The material provided (source data, a classifier script) can be used by students depending on their knowledge in networking or machine learning domains. A novice can start by studying the method and trying to replicate the presented results. Advanced students may start by modifying the method or implementing their own method. Further, they can compare results of their solution with the provided one.

V. CASE STUDY 2: IDENTIFYING A MOBILE DEVICE USING DNS FINGERPRINTING

Network communication of mobile devices is a valuable source of information about the device. From captured communication, it is possible to infer what applications are installed on the system [20], or we can observe user activity [21]. The goal of this study is to observe Domain Name System (DNS) communication and extract specific features from DNS packets that can be used to identify a device. For data processing we employ frequency analysis of DNS queries.

A. Problem Description

DNS system typically translates a domain name, e.g., www.vutbr.cz, to an IP address, e.g., 147.229.2.90. Domain name translation is required for any communication over IP. Before an IP datagram is sent, a DNS client sends a DNS query to the DNS server for domain name resolution. The client forwards the query to its primary DNS server that is either manually or dynamically configured. By observing DNS queries, their types, e.g., A, AAAA, PTR, or requested domains, we can learn much about the device. For instance, if the system supports IPv6, it requests AAAA records in DNS. Also, most of installed applications contact vendor servers for updates or synchronization, which yields in resolution of DNS names like spotify.com, github.com, facebook.net. Similarly to applications, we can identify the operating system by observing queries to DNS domains like android.google.com, xioami.net, etc. Based on the frequency and uniqueness of requested domain names we can create a DNS fingerprint that identifies a unique device.

In our scenario, we will analyze captured DNS communication sent by a set of mobile devices. First, we process a training dataset where we extract selected fields from DNS communication. Based on frequencies of requested domain name lookups we create a DNS fingerprint for each device detected in the dataset. In the second phase this DNS fingerprint is compared with fingerprints obtained from other datasets. If similarity of fingerprints from two datasets is high enough, we can deduce that DNS communication comes from the same source device.

The hypothesis behind the research is that almost every software installed on the device leaves a unique trace in DNS communication. By collecting all these traces we can describe an individual device and distinguish one device from others.

B. Input Data

As source data, we use captured DNS communication. From DNS communication we extract specific values using `tshark`[1] command:

```
tshark -r dns.pcap -T fields -E separator=";"
  -e ip.src -e ip.dst -e dns.qry.type -e dns.qry.name
  "dns.flags.response eq 0 and ip"
```

Using this command, we obtain the following values:

- Source IP address (DNS client's address).
- Destination IP address (DNS server's address).
- DNS query type, e.g., A, AAAA, PTR, etc.
- Requested domain name, e.g., www.googleapis.com.

The source IP address is used to classify DNS requests that belong to the same device. The destination address identifies a primary DNS server that is configured on the device.

Tshark provides a list of tuples {*srcIP, serverIP, query_type, requested_domain*} which are later transformed into a table where each device identified by an IP address will be assigned a vector describing frequency of all DNS queries found in the dataset, see Figure 2. Query type 1 means A record. The frequency vector forms a DNS fingerprint of a device.

C. Method

The naïve solution uses raw frequency vector to build a DNS fingerprint. Then a DNS fingerprint of an unknown device can be compared to a set of DNS fingerprints of known devices.

[1] See https://www.wireshark.org/docs/man-pages/tshark.html [June 2019]

Fig. 2. Creating a frequency table for DNS queries

Based on similarities of two DNS fingerprints we can say how similar these two devices are.

However, raw frequency matching without normalization does not work well with real DNS data. Thus, we applied TF-IDF (Term Frequency–Inverse Term Frequency) method [22]. TF-IDF method includes normalization and weighting. The method is typically used to assign a document to the most similar document taken from a set of documents based on frequency of terms that are found in documents. In case of DNS fingerprinting, we will use domain names to represent TF-IDF terms. For each term in the individual document, Term Frequency value $tf_{t,d}$ is computed using frequency of term t in document d. Normalized Term Frequency is given by the following formula:

$$tf_{t,d} = 0.5 + \frac{0.5 * f_{t,d}}{MaxFreq(d)} \tag{3}$$

where $MaxFreq(d)$ is maximal frequency of any term in document d.

Term frequency suffers from a serious problem: all terms are considered equally important when it comes to assessing relevancy on a query. This means, when a term is present in almost every document, it distorts the ability to distinguish documents based on term frequency. Using *Inverse Document Frequency* (IDF) we can weight frequency by a number of documents in the collection that contains term t. Thus, terms that are present in the small number of documents will be given the higher weight. IDF can be computed using the following formula [22]:

$$idf_t = \log \frac{N}{df_t} \tag{4}$$

where N is a number of all documents and df_t is a number of documents that contain term t. Combination of TF and IDF creates the TF-IDF weighting scheme that assigns a weight to term t in document d by the following equation:

$$tfidf_{t,d} = tf_{t,d} \times idf_t \tag{5}$$

D. Evaluation

In our case, the DNS fingerprint is expressed as a TF-IDF vector that contains weighted and normalized frequencies of domain names found in the dataset. Table III displays frequencies for selected domain names of devices F0 to F3. Column `F0` denotes an unknown device that we will be matched against fingerprints of devices F1 to F3. For example,

TABLE III
FREQUENCY OF DOMAIN NAME LOOK-UPS

	F0	F1	F2	F3
clients4.google.com	1	1	2	16
graph.facebook.com	2	0	9	0
mtalk.google.com	4	0	2	1
pxl.jivox.com	1	1	0	0
www.google.com	6	3	5	8
www.googleapis.com	4	6	8	6

the first domain name `clients4.google.com` has been found in communication of all devices with frequencies 1, 1, 2, or 16 respectively. Domain `pxl.jivox.com` appeared only one time in communication of device F0 and F1.

Based on raw frequencies, we can compute $tf_{t,d}$ and idf_t values for each domain name using equations (3) and (4), see Table IV. Each column TF_{xx} includes a list of TF frequencies

TABLE IV
TF AND IDF VALUES FOR FOUND DOMAINS

	TF_{F0}	TF_{F1}	TF_{F2}	TF_{F3}	IDF
clients4.google.com	0,583	0,583	0,611	1,0	0
graph.facebook.com	0,667	0	1,0	0	0,477
mtalk.google.com	0,833	0	0,611	0,531	0,176
pxl.jivox.com	0,583	0,583	0	0	0,477
www.google.com	1,0	0,75	0,778	0,75	0
www.googleapis.com	0,833	1,0	0,944	0,688	0

for all domain names requested by a given device. By applying IDF frequency using equation (5), we get an TF-IDF value that represents a DNS fingerprint of the device.

Comparison of DNS fingerprints can be evaluated using cosine similarity [23]. The resulting score describes a level of similarity between documents q and d, in our case between two DNS fingerprints:

$$score(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|} \tag{6}$$

$\vec{V}(q) \cdot \vec{V}(d)$ is a dot product of weighted DNS fingerprints $tfidf_{t,d}$. For our input data, similarity score of DNS fingerprint $F0$ towards DNS fingerprints $F1$, $F2$ and $F3$ is in Table V. Based on the score, the most similar fingerprint to F0 seems to be F2.

TABLE V
SCORE VALUES FOR DATASET F0 TOWARDS F1 TO F3.

	F1	F2	F3
F0	0.621	0.767	0.333

E. Recommendation

The presented case study shows how TF-IDF classification can be applied on DNS traffic and used for devices identification using fingerprinting. Students are given two datasets with

DNS communication and scripts that extract DNS communication from PCAP file and converts it into a CSV (Comma Separated Values) format. CSV data can be inserted into SQL database for easier manipulation. Computation of TF, IDF and score values is implemented by a Python script that queries SQL database and computes TF, IDF and score values for database values.

Advanced students may observe the result and add some heuristics to the processing, e.g., apply additional data preprocessing. They can also experiment with other similarity measures, e.g., Euclidean or Jaccard similarity. The full version of both case studies is available at https://github.com/rysavy-ondrej/Ironstone/tree/master/Methods/Detection.

VI. Conclusion and Future Work

AI techniques and especially various methods of Machine Learning gain significant attention in many areas as a promising approach to solve complex problems. The necessity of properly educated engineers that obtain domain knowledge of the particular application field as well as working knowledge of ML methods is evident. Traditionally, ML is taught as a part of specialized courses that explain the theory and various methods in detail, but they often lack the real-world examples for the adequate demonstration of possible applications. We have presented our approach that integrates ML methods as a part of networking courses. The idea is to equip the students with real-world case studies of ML applications in the domain of computer networking in order to increase their understanding of the whole process from the problem identification and formulation to the design of a suitable method and its evaluation. The cases studies are provided with one of the possible solutions supposed to help students increasing their knowledge of ML. In the paper, we have presented two case studies focused on anomaly detection and device fingerprinting.

Further effort aims at the pilot evaluation of this approach in two engineering courses for graduate students. Based on the evaluation, the scope, goal and teaching style for presenting AI methods in engineering curriculum will be adjusted. Also, other cases studies from the networking domain whose solution may involve ML methods will be considered, for instance, the problem of network diagnostics, cyber attack detection, etc.

We believe that working knowledge of AI techniques is inevitable for computer engineers regardless of the field of their specialization. Presented activities aim to support teaching of AI with a strong connection to the application domain.

VII. Acknowledgement

This work is supported by BUT project "ICT tools, methods and technologies for smart cities", FIT-S-17-3964, 2017-2019, and project IRONSTONE - IoT monitoring and forensics", TF03000029, 2016-2019, funded by the Technological Agency of the Czech Republic.

References

[1] J. Manyika, "Digitization, AI, and the Future of Work: Imperatives for Europe," *Briefing Note for EU Tallin Digital Summit*, September 2017.

[2] DIGITALEUROPE, "Recommendations on AI Policy. Towards a sustainable and innovation-friendly approach," Tech. Rep., November 2018.

[3] I. Tuomi, "The Impact of Artificial Intelligence on Learning, Teaching, and Education: Policies for the Future," Publications Office of the European Union, Tech. Rep. EUR 29442 EN, 11 2018.

[4] ACM and IEEE-CS, "Computer Engineering Curricula 2016. Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering." ACM/IEEE Computer Society, Tech. Rep., December 2016.

[5] H. Topi, S. A. Brown, B. Donnellan, B. C.Y.Tan, H. Karsten, J. A. Carvolho, J. Shen, and M. F. Thouin, "MSIS 2016. Global Competency Model for Graduate Degree Programs in Information Systems." ACM/AIS, Tech. Rep., December 2017.

[6] ACM and IEEE, "Cybersecurity Curricala 2017. Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity," ACM/IEEE-CS/AIS-SIGSEC/IFIP WG 11.8, Tech. Rep., December 2017.

[7] J. Kurose, J. Liebeherr, S. Ostermann, and T. Ott-Boisseau, "ACM SIGCOMM Workshop on Computer Networking:Curriculum Designs and Educational Challenges," *ACM SIGCOMM Computer Communications Review*, vol. 32, no. 5, 2002.

[8] M. O. Popescu and C. L. Popescu, "Trends in evolution of electrical engineering curriculum," in *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*, Oct 2017, pp. 1–5.

[9] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine Learning for Networking: Workflow, Advances and Opportunities," 2018.

[10] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," 2008.

[11] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems," *IEEE Communications Surveys and Tutorials*, 2017.

[12] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Computer Communication Review*, 2006.

[13] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, "Unsupervised traffic classification using flow statistical properties and IP packet payload," *Journal of Computer and System Sciences*, 2013.

[14] J. P. Early, C. E. Brodley, and C. Rosenberg, "Behavioral authentication of server flows," in *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 2003.

[15] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced Intrusion Detection System," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2016.

[16] L. Dhanabal and D. S. P. Shantharajah, "A Study On NSL-KDD Dataset For Intrusion Detection System Based On Classification Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, 2015.

[17] Z. Shelby, K. Hartke, and C. Bromann, "*The Constrained Application Protocol (CoAP)*," IETF RFC 7252, June 2014.

[18] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: A statistical anomaly approach," *IEEE Communications Magazine*, 2002.

[19] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, 2007.

[20] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting mobile devices using personalized configurations," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 4–19, 2016.

[21] G. Chittaranjan, J. Blom, and D. Gatica-Perez, "Who's who with big-five: Analyzing and classifying personality traits with smartphones," in *2011 15th Annual International Symposium on Wearable Computers*, June 2011, pp. 29–36.

[22] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[23] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.