Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Traffic classification at the radio spectrum level using deep learning models trained with synthetic data

Tom De Schepper | Miguel Camelo | Jeroen Famaey | Steven Latré

[1]University of Antwerp - imec, IDLab,Department of Computer Science, Antwerp, Belgium

Correspondence
Tom De Schepper, University of Antwerp - imec, IDLab - Department of Mathematics and Computer Science, Sint-Pietersvliet 7, 2000 Antwerp, Belgium. Email: tom.deschepper@uantwerpen.be

Abstract

Traffic recognition is commonly done using Deep Packet Inspection or packet-based approaches. However, these methods require the listening device to be part of the network and raise privacy concerns. Traffic recognition models that operate directly at the spectrum level could, for instance, be used for smart spectrum management. To this extent, we present such an approach using deep learning methods. In particular, we present a Convolutional Neural Network architecture that forms the basis for prediction models to recognize different transport protocols, burst traffic with different duty cycles, and different transmission rates. These models are trained with pure synthetic data to lighten the burden of data collection. As such, we validate recent successes in the area of robotics in the context of wireless networks. We compare the performance of two different datasets that contain spectrum images in either time or time-frequency domain. Our evaluation shows that using time-domain data results in an accuracy of at least 96 % across all models. Time-frequency information improves this accuracy even further. Furthermore, a validation with real-life data shows that it is still possible to discriminate between different transmission rates with an accuracy of around 87 %, while the detection of duty cycles and transport protocols takes place with accuracies of, respectively, around 73 and 78 %. Finally, we also present a small-scale real-life prototype.

KEYWORDS:
Traffic recognition, Spectrum Management, Deep Learning, Convolutional Neural Networks

1 | INTRODUCTION

Traffic classification aims to identify different patterns of information in network traffic streams. It has been around since the beginning days of the internet and knows many applications [1,2,3]. The typical use cases are network management (e.g., providing Quality of Service (QoS) or billing) by Internet Service Provider (ISPs) and intrusion or anomaly detection in the area of network security. More recent application domains are smart spectrum management and Cognitive Radio (CR). Here, efforts are being made to enhance the coexistence of different wireless networks and devices. This coexistence is being pressured by the drastic changes to the Internet in the last decade, in terms of the number of connected devices, the amount of traffic, and the availability of different communication technologies [4,5]. For instance, the number of connected devices has surpassed 18 billion and will reach 28.5 billion by 2022 [4], while the set of available communication technologies will be further expanded with the releases of new IEEE 802.11 (Wi-Fi) standards (e.g., IEEE 802.11ax and IEEE 802.11ay) and 5G technologies) [5]. There is a strong need for

the efficient and intelligent use of spectral resources [6]. This is due to the fact that these devices and technologies typically operate next to each other at identical or overlapping physical locations, often competing for the same finite and limited radio spectrum resources, which leads to significant QoS degradations and performance loss [7]. Furthermore, the aforementioned growth has led to radio spectrum scarcity since both licensed and unlicensed spectrum bands are getting extremely crowded [7]. In order to have such adaptive management systems, it is crucial that these systems have access to accurate and real-time information of the state of the wireless spectrum.

However, existing traffic recognition approaches are unfit for this challenge. One of the main reasons is that existing approaches operate on a packet-level, which typically requires that the listening device is part of the corresponding network [1,2,3]. This makes it infeasible to monitor and acquire information from neighboring networks or technologies. Furthermore, the default method used by industry is still Deep Packet Inspection (DPI). While this method is very accurate, it requires a lot of computational power, is very intrusive (i.e., towards privacy), and often requires manual signature maintenance [1]. The intrusive nature of DPI is especially cumbersome in light of the recent global focus on data transparency and privacy regulations (e.g., General Data Protection Regulation (GDPR)). As an alternative, statistical and Machine Learning (ML) approaches have been proposed which do not require packet inspections but are based on captured packet traces, typically at the Internet Protocol (IP) level [1,8]. These methods still require access to the network in order to collect packet traces and are typically available for wired networks only. As such, it is clear that there is a need for new approaches that can operate within the wireless context of today and respect the privacy requirements of modern applications and users. In this paper, we explore such an approach that aims to detect traffic patterns at the spectrum level. The detected information can be exploited for network and traffic optimization such as smart spectrum management. In particular, we use Deep Learning (DL) methods.

A critical issue when constructing ML models in general, is the collection of sufficient amounts of quality data. Additionally, given the very dynamic characteristics of the different wireless environments, ML models need to be robust to handle this challenging context. Recent breakthroughs in the area of robotics have shown how models can be successfully trained purely with synthetic data [9,10,11]. By using a persevering form of randomization, the so-called reality-gap between simulators and reality can be bridged, and the models learn to consider reality as just another adaptation as the model only captures the bare essential [10,11]. This technique has a strong potential as it allows for more robust models and an easy, safe, and reliable manner to collect large amounts of data. In this manuscript, we present a first step towards the use of this principle within the context of wireless networks.

The contributions of this approach are five-fold. First, we present the models to recognize Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic, burst traffic with different duty cycles, and different types of TX rate. A Convolutional Neural Network (CNN) architecture forms the base for these different traffic recognition models. Second, we propose a novel data randomization approach to generate large amounts of (synthetic) data, combining two state-of-the-art simulators. Third, we explore and compare two different approaches to represent the sensed spectrum as the input of our models: time and time-frequency image representations. Fourth, we evaluate our synthetically trained models with real-life data, as such showing the applicability of domain randomization in the context of wireless networks. Fifth, we present a small-scale prototype where we implement one of the models to demonstrate the deployment in a real-life setting.

The remainder of this paper is structured as follows. We start by giving an overview of the state-of-the-art in Section 2. We introduce our DL architecture and traffic recognition models in Section 3, while presenting the data collection framework in Section 4. Next, in Section 5, we show an evaluation of the different models using test data, real-life data, and our prototype implementation. Finally, we draw conclusions and indicate topics for future work in Section 6.

2 | RELATED WORK

In order to increase the spectrum efficiency and to cope with neighboring networks, we propose to recognize traffic patterns at the level of wireless radio signals. However, traditionally, traffic recognition takes place at higher layers of the network stack. To this extent, we discuss these traditional methods in Section 2.1. Next, in Section 2.2, we take a look at the efforts made in the area of CR. Finally, we also provide a summary in Section 2.3.

2.1 | Traditional traffic recognition approaches

Traffic recognition commonly takes place at gateways or routers in a (wired) network aiming to identify individual traffic streams or applications entering or exiting the controlled network environment. The typical use cases are network management (e.g., providing QoS or billing) by ISPs and intrusion or anomaly detection in the area of network security [1,2]. Two main approaches exist: DPI and methods based on (IP) packet traces. Historically, DPI has long been the default approach where information is extracted from the headers and payload of individual packets, typically in a rule-based system. Especially towards network security, this can take place at different layers of the network stack (varying from the MAC to application layer) [2]. While DPI methods are very accurate, they require a lot of computational power, are very intrusive (i.e., towards privacy), and often require manual signature maintenance [1]. The intrusive nature of DPI is cumbersome in light of the recent global focus on data transparency and privacy regulations (e.g., GDPR). Furthermore, DPI can not always cope with the encryption of many modern end-to-end services. To this extent, recent work leverages the trade-off between encryption and privacy on one hand, and the functionalities and information exposed by DPI on the other hand [12]. The authors propose a set of novel protocols and encryption schemes that make it possible to perform DPI directly on the encrypted traffic, without compromising the privacy of the captured data.

As an alternative to DPI, statistical and ML-based methods have been proposed that do not require packet inspections but are based on captured packet traces, typically at the IP level [1,8,3]. These methods are based on traffic flow statistics like packet sizes, flow durations, inter-packet times, source and destination ports, and IP addresses [1,13]. Both clustering (e.g., Expectation Maximization or K-Means), supervised learning (e.g., Naive Bayes or Genetic Algorithms), and semi-supervised learning (e.g., a combination of clustering and label mapping) techniques have been applied previously [13]. For instance, semi-supervised learning has been proposed to cope with zero-day applications, previously unknown to the traffic classification systems [14]. Overall, these methods are capable of handling encrypted traffic, offer a lower computational cost than DPI methods, while still acquiring a rather high accuracy for flow classification (up to 99 %) [1,13]. Lately, deep learning approaches have also been proposed [8,15]. Lotfollahi et al. propose a combination of a stacked autoencoder with 5 fully connected layers and a one-dimensional CNN with as input IP layer packets [8]. The approach is capable to identify applications with an accuracy of 98 % and traffic characteristics with a precision of 93 %. Furthermore, a CNN architecture is also presented by Wang et al. to detect malware traffic [15]. The model is trained on images, that are acquired by converting packets (pcap to idx format). Open issues include, among others, the questionable performance under non-perfect circumstances (e.g., packet loss, jitter, and packet fragmentation), while also only a limited amount of accurately labeled training data sets are available [1,13]. Note that these available datasets only include data captured on wired links.

2.2 | Cognitive radio approaches

The principle of CR has been introduced, comprising a large number of different approaches, to allow the coexistence of different technologies (both licensed and unlicensed) and networks over the same spectrum [6]. Within this domain of CR, the detection and classification of wireless radio signals are important to optimize spectrum usage. So far, the focus has been mostly on the recognition of modulation schemes and technologies. The detection of different technologies is typically done by exploiting key differences in, for instance, the channel access methods of the different technologies [16,17]. Typical methods that have been applied are likelihood-based and feature-based, using high-order statistics features such as moments, cumulants and cyclic cumulants. For instance, Shi et al. propose a method to differentiate single carrier from Orthogonal Frequency Division Multiplexing (OFDM) signals, using the fourth order cumulants as features [16]. Karami et al. show how to discriminate between spatial multiplexing OFDM and Alamouti-coded OFDM for Multiple-input and Multiple-output (MIMO) systems by relying on the second-order signal cyclostationarity [17]. Furthermore, they are also capable of distinguishing between GSM and Long-Term Evolution (LTE) transmissions [18]. Finally, Liu et al. present an approach that uses the differences in Received Signal Strength Indicator (RSSI) distribution at Sub-Nyquist sampling rate to successfully recognize different technologies (e.g., Wi-Fi or LTE) [19].

Recently, deep learning techniques have gained momentum and are applied to classifying wireless signals. Schmidt et al. apply a CNN architecture to discriminate, with an accuracy of 95 %, between 19 different variants of modulation types and symbol rates within the 2.4 GHz band [20]. Similarly, Jeong et al. also apply a CNN to recognize different modulation types based on spectrograms, while Rajendran et al. apply a Long short-term memory (LSTM) model that can classify 11 modulation types with an accuracy of 90 % [21,22]. Both O'Shea et al. and Kulin et al. show how CNNs significantly outperform expert learning systems for the task of radio signal classification [23,24]. They both compare multiple differently structured CNNs that are trained on processed In-phase and Quadrature (IQ) samples. Furthermore, a CNN has been developed to detect different interfering

sources for an IEEE 802.15.4 sensor network, using RSSI traces[25]. The approach allows to detect with an accuracy of 93 % Wi-Fi beacons, Wi-Fi video streaming, Wi-Fi file transfer, iBeacon, and microwave oven interference in the 2.4 GHz band. Lately, Testi et al. have also shown that is it possible to use ML techniques to distinguish YouTube form WhatsApp traffic in a real-life setting[26]. The best results were achieved with a Neural Network (NN), achieving an accuracy of above 90 % under different signal to noise ratio (SNR) levels. Finally, O'Shea et al. have shown that recurrent neural network models can be used to recognize high-level radio protocols by using sequences of raw IQ samples[27]. In this work, the authors were able to classify different types of applications, such as streaming, chat, browsing, and file transfers, among others. They achieve up to 83% of accuracy using an input of size 65536 IQ samples, which were sampled at a bit rate of 1Mbps.

2.3 | Summary

We have shown that traffic recognition is historically always performed at the higher layers of the network stack by DPI and methods that are based on IP packet traces. These methods are, in general, very accurate and are actively being used by (e.g., by ISPs) but are only employed on wired networks. Furthermore, the popular DPI method is computational intensive and does not respect the privacy requirements of modern users or applications. In contrast, efforts have been made recently in the area of CR, often based on deep learning and in particular using a CNN architecture. While most approaches target the recognition of wireless technologies and modulation schemes, recent work has directed attention towards identifying specific traffic patterns as well. However, most of these works target only specific and very distinct types of traffic (e.g., differentiating YouTube from WhatsApp) and more generic traffic classes or transport protocols have not yet been considered. Finally, recent approaches using raw spectrum data with sequential models, such as LSTM networks, suffer from the lack of performance as they can not exploit the parallelization capabilities of moderns GPUs. Moreover, it is also harder to train such models, compared to CNN architectures, which can have a similar or even better performance than LSTMs[28]. Overall, we can conclude that applying traffic recognition directly at the level of the wireless spectrum is still a completely open research challenge.

3 | TRAFFIC RECOGNITION MODELS

In this section, we start with the introduction of a formal problem definition of recognizing traffic patterns at a spectral level. Afterwards, we discuss how the spectrum data is used and define the necessary labels. We round up this section by presenting the shared CNN architecture and the three different prediction models.

3.1 | Problem definition

Our goal is to explore if it is possible to detect traffic patterns directly in the wireless spectrum. We propose to use a Supervised Learning (SL) approach to see if patterns can be detected at all. The following problem formulation is defined:

Let $X = \{x_1, x_2, \dots, x_N\}$ and $Y = \{y_1, y_2, \dots, y_N\}$ be the sets of N examples and their corresponding labels, respectively, where $x_i \in X$ and $y_i \in Y$ for all $i \in [N] := \{1, 2, \dots, N\}$. In SL, the goal is to learn a mapping from X to Y given a training set of pairs (x_i, y_i), where y_i is the label of the ith example x_i. In other words, given the sets X and Y, SL tries to find a function f such that $y = f(x)$.

Within the scope of performing traffic recognition using spectrum data, the problem is to find a function f that given a representation of the spectrum $x_i \in X$ (e.g., raw IQ samples or Short Time Fourier Transform (STFT) of the IQ samples), is able to predict $y_i \in Y$. In this case, Y is a set of labels that represent the properties of the transmitted traffic that we are trying to identify. These traffic properties can, among others, be the employed transport protocol, transmission rate, or the traffic pattern of the application (e.g., burst or constant). In the next sections, we propose to use DL architectures to solve the above-described problem. In particular, we employ CNN models to build the function f through a combination of many less complex functions that are connected hierarchically[29].

3.2 | Input spectrum representation and traffic labels

Before designing and implementing a CNN for traffic recognition tasks using spectrum data, we need to define the type of input data that represents the radio spectrum. A popular method within the area of CR is the use of IQ samples. IQ refers to two

sinusoids in which a radio signal can be decomposed. Those two sinusoids have the same frequency and are 90° out of phase (hence *in quadrature*). By convention, the I signal is a cosine waveform, and the Q signal is a sine waveform. For completeness, the mathematical formulation for this can be seen in the following equation:

$$s(t) = i(t) + q(t) = I(t)\cos(2\pi f_c t) + Q(t)\sin(2\pi f_c t)$$

With $s(t)$ the original transmitted signal, $i(t)$ the in-phase component, $q(t)$ the quadrature component, and f_c the center frequency of the carrier. This approach is widely used due to its relatively simple mathematical operations and its flexibility to generate any modulation scheme by combining the appropriate I(t) and Q(t) values. Note that IQ signals are always modulated based on the amplitude.

Typical CR tasks that make use of IQ samples are technology and modulation identification, as highlighted in Section 2.2. These tasks discriminate between different classes (e.g., different technologies) using features that are visible in short periods (at most hundred of microseconds)[23,24]. For instance, let us consider the task of modulation recognition, as presented by O'Shea et al.[23]. As input, 128 IQ samples are used that represent 128 us of the signal at a sampling rate of 1 million samples/s. However, in contrast to these related tasks, traffic recognition is based on traffic features that are only visible in longer periods of time (at least in a magnitude of hundreds of milliseconds). If we would consider the same sampling rate as in the aforementioned example, representing 0.5 s of spectrum data for traffic recognition would require 0.5 million of IQ samples. However, implementing ML models that utilize this input size is impossible, due to the enormous amounts of memory, computational resources, and storage required in order to train and run such models.

With the need of having a broader view of the spectrum in time, while keeping the input data as small as possible, it is clear that directly using IQ samples is not an option. To this extent, we propose the following data transformation: given a time window w (in seconds) and raw spectrum data as input (i.e., IQ samples), the amount of data collected during the interval w is plotted and saved as an image (in compressed format). In this work, we use a window size $w = 0.5$ s (experimentally determined), while the generated images are RGB (normalized between 0 and 1) with a size ([height,width]) of [656,874] pixels. Each pixel represents an averaged value of the transmission power observed at that particular moment. Furthermore, images are saved in the Portable Network Graphics (PNG) format. As a result of this data transformation, the size of the images, generated during this research, varied in a range between 3.3 KB and 114 KB. In general terms, the required storage for the image dataset is less than 1 % of the dataset based on raw IQ samples. In Section 5, we provide more details on the resulting datasets. Intuitively, the idea of converting IQ samples to images as input for DL models make sense, due to the high success of applying DL techniques on pixel inputs[30,31,32].

Finally, in order to generate labels for the input dataset, we focus on the recognition of three different properties of the generated traffic: the transport protocol (TCP versus UDP), the traffic pattern (constant versus 3 types of burst traffic with a duty cycle of, respectively, 25, 50, or 75 %), and the rate at which the traffic was transmitted (100 Kbps, 1 Mbps, 10 Mbps, or 50 Mbps). These three different traffic properties are examples of characteristics that can be used to identify distinct traffic flows, applications, or classes of traffic. For instance, discriminating between TCP and UDP traffic can help in detecting applications or flows that require reliability. This information can also help management algorithms to cope better with the significant different behavior of those two protocols (e.g., TCP fairness behavior). Furthermore, recognizing burst traffic with different duty cycles can help to detect recurrent patterns of free transmission time. When combined with other features, it can also be used to identify different classes of traffic and applications (e.g., video streaming versus email traffic or file downloads). Finally, detecting the individual transmission rates can help in understanding the bandwidth requirements of flows or applications, while it is also possible to better detect the amount of free bandwidth available at certain frequencies. Each generated image is associated with a label for each of the three properties. More details on the process and framework to generate the data are presented in Section 4.

3.3 | Convolutional Neural Network for traffic recognition

In order to exploit the new representation of the input data as images, we design and implement three different DL models based on a CNN architecture. CNN architectures are specifically designed for, and have proven their worth in, many computer vision applications[33]. Moreover, they have become the de facto standard in the area of computer vision[34]. Under this assumption, the forward function (i.e., moving the data from the input to the output side through the neural network) is more efficient and its computational complexity for learning is lower than traditional Deep Neural Networks (DNNs) based on fully-connected layers. This is because a CNN connects the neurons of a given layer, called a convolutional (or conv) layer, with only a few neurons of the next layer, reducing the number of parameters that need to be trained. Furthermore, these kinds of DNNs have been shown,

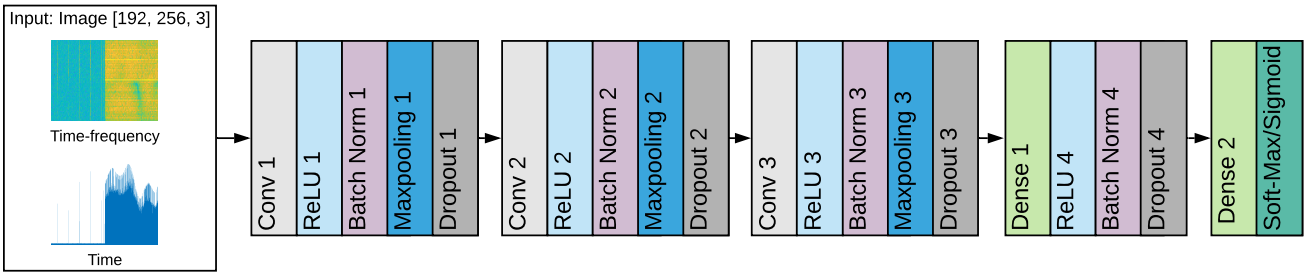


FIGURE 1 CNN architecture for traffic recognition

as indicated in Section 2.2, to perform well in other related tasks such as modulation recognition. These models work either directly with raw spectrum data as input (e.g., IQ or Fast Fourier transform (FFT) samples) or with image representations of that raw spectrum data[23,21].

We design a baseline architecture that is shared by all the models. In total, there are 6 models. For each classification task (three in total) there is a model for both time-domain data and time-frequency domain data. Overall, the architecture consists of 6 different sequential blocks of layers, as can be seen in Figure 1.

The first block is the input layer. This layer takes an image of size 192x256x3 as input, which is representing either a STFT (i.e., a spectrogram), or the raw IQ samples in magnitude (i.e., the time series plot). The size of the first dimension (i.e. vertical axis in the image) corresponds with the frequency resolution used to extrapolate/interpolate the FFT samples used to create the spectrograms, or the float values used to interpolate the changes of the magnitude of the signal. The size of the second dimension (i.e. horizontal axis in the image) is the time resolution used to interpolate the power spectral density or the IQ sample magnitudes in a given time window. For instance, using the input size in our work, if the time series plots are representing 0.5 seconds of the magnitude of the IQ values, which is equivalent to an array of 10 M floating points at a sampling rate of 20 Msps, then these points are processed (interpolated) to get 256 values with a magnitude in the range $[0, 1]$ in steps of 0.0052 units (192 values). This interpolation process applied on the raw data in memory to create the images, which is performed by a PNG compression-filtering algorithm in our implementation, allows us to create ML models that exploit the capabilities of CNNs for finding local patterns that are relevant everywhere and run in commodity hardware as we mentioned in Section 3.2. Moreover, the size of the third dimension indicates that the images are RGB (truecolor images), which allows us to see the power spectral density (PSD), in spectrograms, and both the magnitude and distribution of the signal in the time domain. Using black and white and images would reduce this dimension from three to one. Finally, notice that although increasing the values of the two first dimensions allows achieving high accuracy in recognizing spectrum features, e.g., short ACKs can be detected with more probability to discriminate TCP versus UDP, or real PSD values in technologies that use a large number of FFT points during sampling, this also requires increased computational power to train and use the resulting neural network.

The following three blocks, each consisting of 5 different layers, are used to detect features. These blocks are built up by the following components: first, a convolution layer that contains a number of feature maps (also known as kernels or filters). These kernels are weighted matrices that are trained to detect and highlight certain features (i.e., object shapes) in the data. The number of kernels increases throughout the convolution layers of the three blocks, in order to step-by-step learn more detailed features. Next, is a Rectified Linear Unit (ReLU) activation function, followed by batch normalization and a Maxpooling function. ReLU is these days the most common activation function, as it converges fast, is cheap to compute (in terms of computational resources), and does not suffer from the vanishing gradient problem in deep neural networks[35,36]. The pooling layer downsamples the given input, in order to further reduce the number of parameters, while also selecting the most relevant (hence the Maxpooling) inputs for the next block [37]. The fifth dropout layer is employed to increase generalization, as it randomly disables some of the neurons depending on the dropout rate (as shown in Table 2). This forces each of the remaining neurons to learn different features, increasing overall robustness. Each of the three blocks of layers is capable of detecting features at a certain level. Stacking multiple of such blocks (i.e., convolutional layers) allows for more complex features to be detected [37]. Note that, by stacking different layers, the chance of overfitting increases, as well as the amount of computational power and time required to train the models. The chance of overfitting can be reduced by using large amounts of data or generalization techniques such as the aforementioned random dropout.

Model	Data Domain	Classification Task	# Kernels	Kernel Size (all Conv layers)	Pool size (all MaxPool layers)	Dense 1 # Neurons	Dense 2 # Neurons
CNN 1	Time	Traffic Pattern	Conv 1 : 16 Conv 2 : 32 Conv 3 : 64	5 x 5	2x2	512	4
CNN 2	Time-Fequency						
CNN 3	Time	Transport Protocol					2
CNN 4	Time-Fequency						
CNN 5	Time	Transmission Rate					4
CNN 6	Time-Fequency						

TABLE 1 Shared hyperparameters among all the models

Dropout Rate	CNN 1	CNN 2	CNN 3	CNN 4	CNN 5	CNN 6
Layer 1	0.1	0.6	0.7	0.6	0.7	0.5
Layer 2	0.3	0.5	0.5	0.3	0.4	0.4
Layer 3	0.6	0.5	0.5	0.3	0.4	0.4
Layer 4	0.6	0.3	0.3	0.1	0.2	0.2

TABLE 2 Specific dropout rate per model

Furthermore, the last two blocks are used for classification. The fourth block is a dense layer with, once again, ReLU as the activation function. This layer also transforms the output of the last block of convolutional layers into a one-dimensional vector. Afterwards, this output is delivered to the last block, which is composed of a dense layer where the number of neurons equals the number of predefined labels for classification. If it is a binary classification (e.g., to distinguish between TCP and UDP), a sigmoid activation function is employed to map the input to the predefined labels. Otherwise, for the case of multi-class classification (e.g., for the transmission rate and duty cycle recognition models), a soft-max layer is used.

Figure 1 shows an overview of the resulting architecture, while Table 1 and Table 2 list the parameters used on each layer per model. Some specific parameters for the optimizer (e.g., the optimizer, learning rate, and batch size), for each layer (e.g., the number of filters, pool size, and dropout rate), and optimizer training, were determined using hyperparameter swapping. In total, the constructed CNN models have 25.2 million trainable parameters. We trained the models during 100 epochs using the Adam optimizer with a learning rate of 0.0005, cross-entropy loss function, and mini-batches of 128 RGB images[38]. These images were resized to [height,width] = [192,256] pixels due to memory constraints in the Graphics Processing Unit (GPU). We implemented our models using the Keras library and TensorFlow as the back-end[39,40].

4 | DATA COLLECTION FRAMEWORK

One of the key challenges faced for the construction of any ML model, is the collection of sufficient amounts of quality data, needed for training and validation. In Section 2 we have mentioned how traditional traffic recognition approaches are typically operating or being trained, with wired packet data. Furthermore, as existing CR approaches are focusing on technology and modulation recognition, no accurately labeled datasets for traffic recognition in the wireless spectrum are currently available. To this extent, we present a new data collection framework in this section. We start by giving a motivation for our approach and, afterwards, present the architecture and features of the framework.

4.1 | Motivation

A key concern is that the characteristics of wireless networks (e.g., interference, latency, or topology) tend to differ significantly across different environments. This means that it is not straightforward to train a model in a general fashion, applicable to a wireless context. Typically, this requires large volumes of training and validation data, gathered at different physical locations. Acquiring these amounts of data is cumbersome and time-consuming. To this extent, the use of simulation environments is very appealing as it allows to collect potentially large amounts of (training) data in an easy, safe, and reliable manner without

the need to perform physical experiments. As such, limiting the effort of data collection. However, the behavior of agents and models trained in simulator environments are often specific to the characteristics of the simulator[9,10]. This due to the errors and abstractions made by modeling the complex real world.

Recently, it has been proved that this so-called reality cap, between the real-world and the simulators, can be bridged using a persevering form of randomization. This technique, called domain randomization, trains learning models based only on (low-fidelity) simulated data by randomizing all non-essential aspects of the simulator[9,10]. One of the core ideas behind this approach is that by training on a wide enough variety of unrealistic procedurally generated samples, the learned models will generalize to realistic scenarios[10]. In other words, the model should consider reality, as just another adaptation as the model only captures the bare essentials[11]. As such, more robust models have been constructed that cope with more dynamic behavior in the real world[9].

Up to now, the technique has mostly been used in the area of robotics and self-driving vehicles where typically cameras and computer vision techniques are being used for object recognition. One of the main applications so-far has been robot grasping, where a robot needs to be trained to pick-up or place certain objects. Tobin et al. have shown that their grasping model, trained entirely using non-realistic generated objects in simulation, can achieve high success rates: a successful grasp was observed for 96 % upon the first 20 samples, while in general a success rate of 92 % was achieved on realistic objects on the first attempt[10]. Furthermore, applications and models for object recognition (e.g., detection of other cars in an autonomous vehicle use case) have shown similar success rates. Here, aspects such as lightning, pose, object textures, and colors are randomized in non-realistic ways to force the neural network to learn the essential features[41,11]. A key observation is that these models could be used in real-world environments without any retraining.

It is clear that domain randomization is an emerging and promising research area. As such, we will further explore this principle in the significant different application domain of wireless networks. We want to investigate if by using this approach, a robust traffic recognition model can be trained that can cope with the dynamic wireless context. In the next section, we will present a data generation architecture, as the first step towards a full data randomization approach.

4.2 | Architecture

In Section 3.2 we have discussed that images, generated based on captured IQ samples, are used as input data for the different models. This means that a simulation environment is needed that can generate such IQ samples at the physical layer of the Open Systems Interconnection (OSI) stack. Furthermore, since we want to recognize traffic patterns, the simulator should also contain the logic of the higher layers (e.g., transport layer) of the OSI stack. Unfortunately, no simulator currently exists that extensively covers all different layers of the network stack. As such, we combine two state-of-the-art solutions that each cover a part of the network stack. We use the discrete-event network simulator NS-3 (version 3.29) to cover the traffic generation and the higher layers of the stack[42]. This includes the transport layer that, for instance, contains TCP rate control mechanisms. Regarding the lower layers, we consider the Matlab toolbox, in particular, to address the physical layer, as the toolbox can generate wireless signals and Radio Frequency (RF) spectrograms (i.e., IQ samples)[43]. We make use of the WLAN toolbox version 2.0 integrated with Matlab R2018b.

The overall architecture of the proposed framework is depicted in Figure 2. As can be seen, there are three main components. First, there is the experiment generator that receives as input a scenario description in JSON format. This scenario description contains for several selected parameters a list of values to be considered in the simulations. The tunable parameters are depicted in Table 3. Note that this formulation is easily extendable to include other features and parameters (e.g., MIMO, packet aggregation, or more complex applications). The experiment generator (1) generates an experiment setup for each unique combination of all parameter values. For instance, consider the (theoretical) scenario where two parameters A and B are specified, with the parameters having, respectively, the following values: (A1, A2) and (B1, B2, B3). The experiment generator will in total create six unique configurations (A1B1, A1B2, A1B3, A2B1, A2B2, A2B3). Note that this experiment generator can potentially create a very large amount of combinations, under a very large parameter space, making it infeasible to verify all unique combinations. While this is not the case for the work currently presented in this chapter, future work could explore surrogate modeling techniques to cope with these massive data spaces.

Next, based on the generated experiment setup, the NS-3 simulator constructs the desired network topology with the specified technology standard and the number of stations and access points (APs). During the length of the simulation, the specified traffic is generated through this network, while stations can move (if applicable) based on the selected mobility pattern. Furthermore, a log file is generated that contains all the transmitted packets and transmission information needed to generate the transmitted

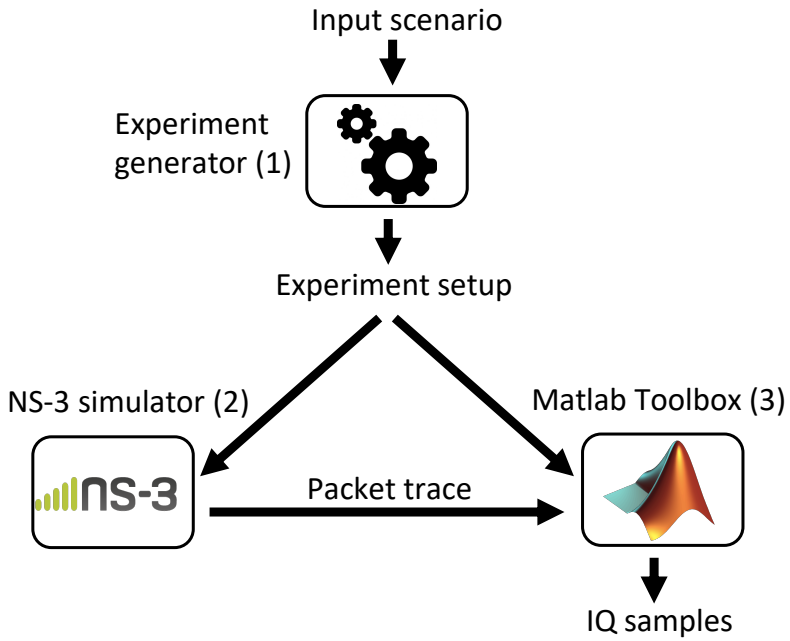


FIGURE 2 Architectural overview of the data generation framework

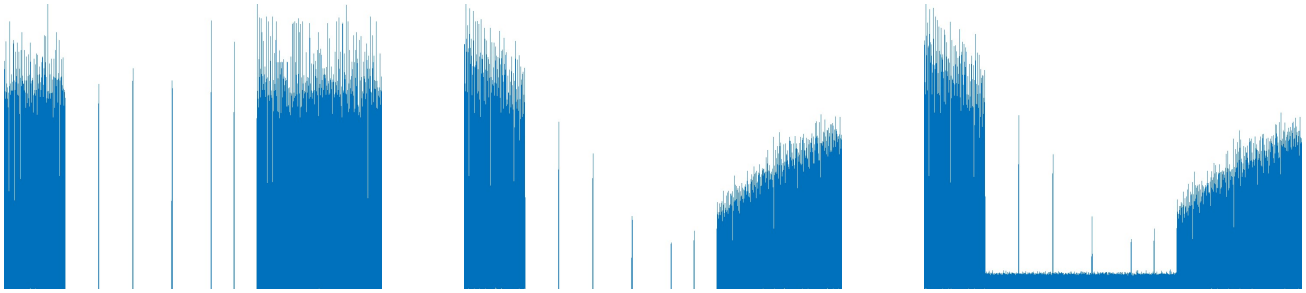
Considered parameters	
Simulation	simulation length
Network topology	number of the nodes, their locations, and mobility patterns
Traffic descriptions	number of flows, transport protocol, traffic patterns, and rates
Technology (Wi-Fi parameters)	Modulation and Coding Scheme (MCS), channel width, standard, and guard interval

TABLE 3 Overview of tunable parameters for data generation

waveform in Matlab. This log can be seen as a regular packet dump (cf. a pcap file), but augmented with the following information: the specific type of packet (e.g., beacon, association request, data), technology specific information (e.g., for Wi-Fi: STBC, PSDU length, AMPDU), and the MCS value.

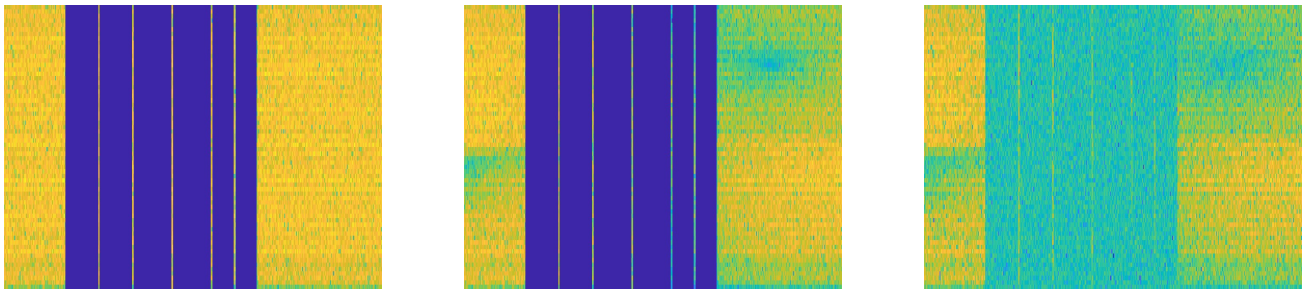
Afterwards, Matlab (3) performs the following three steps: first, a MAC packet is generated per line of the previously generated trace file in combination with the information provided. This MAC packet is used to generate a waveform that is compliant with the 802.11 standards. The waveform is expressed as an array of complex numbers that represent the IQ samples of the signal. In this work, we assume that there is only one transmitter that is transmitting at a given time to avoid cross-interference or the mixing of different types of traffic in a given time window. However, this framework allows generating a waveform representing multiple transmitters in a given time window by synchronizing, using the timestamps of the packets inside of the pcap files, and adding the waveform generated by each transmitter individually. In this case, at the receiver side, it is required to 1) enumerate all transmitters in a given time window, and 2) separate the different signals to be further processed individually[44,45], i.e., decomposition of the original image and aggregation of transmission from the same transmitter. In the case of receiving signals resulting from interference among multiple transmitters, which often result in failed packets, recent work in technology recognition/identification[46,47] will allow filtering out those failed transmissions to provide successful not-interfered transmissions, and even apply the proposed algorithm on multiple technologies by separating the radio signals by technology.

After each packet is generated, the IQ samples are stored in binary format in a file. Note that the IQ samples are, at this stage, free of noise and do not include any channel effects. Second, the generated IQ samples are augmented by passing them through both a modeled fading channel (according to the 802.11 standards) and an Additive white Gaussian noise (AWGN) channel. The latter adds white Gaussian noise to the signal, with an SNR from 0 dB to 30 dB in steps of +3 dB. This means that



(a) Original signal (no channel or noise effects) **(b)** Signal with channel effects **(c)** Signal with channel and noise (15 dB) effects

FIGURE 3 Three images in time domain with different channel and noise effects



(a) Original signal (no channel or noise effects) **(b)** Signal with channel effects **(c)** Signal with channel and noise (15 dB) effects

FIGURE 4 Three images in time-frequency domain with different channel and noise effects

per original IQs sample, an additional amount of 12 IQ samples are created: 1 with only channel fading effects, and 11 with channel fading and noise (each with a different SNR). This is done in order to randomize the spectrum conditions. As such, following one of the main principles behind the domain randomization technique, as discussed previously. Third, based on the previously created IQ samples, we plot them in time (amplitude of the IQ samples) and time-frequency (by applying the STFT on the IQ samples) domain and save them to disk. This leads to two different input datasets, which are compared in Section 5. Note that each created image represents a traffic snapshot of size $w = 0.5$ s, which is equivalent to 0.5 million of IQ samples. To clarify, imagine an experiment length of 5 s. Since we take snapshots of 0.5 s, 10 (experiment length divided by snapshot size) $\times$ 13 (the clean image + the additional ones with channel fading and noise effects) images are created for the specific scenario, for both time and time-frequency domain. Additionally, we want to highlight that the presented approach is only a first step towards a full domain randomization approach, as a complete domain randomization approach also considers directly changing the classification models and the use of non-realistic randomized data.

To illustrate the generated datasets, we depict for both the time and time-domain dataset three images in, respectively, Figures 3 and 4. The images are one of the snapshots made for a scenario with a UDP traffic flow of 10 Mbps and a duty cycle of 75 %. Each figure is composed of three images that respectively the different steps in the data adaptation process, as described above. The top image shows the created snapshot of the original signal based on pure IQ samples). Furthermore, the middle image shows the effect of adding 802.11 channel effects to the signal, while the adding of noise to the signal, using an SNR of 15 dB.

5 | EVALUATION AND DISCUSSION

In this section, we evaluate the accuracy of the three traffic recognition models, while also investigating the use of the models in real-life settings. First, we discuss the details of the generated datasets for the training and validation of the presented models.

Model			Accuracy			textbfPrecision	Recall
			Train	Validation	Test	Test	Test
CNN 1	Traffic Pattern	Time	0.999	0.962	0.962	0.960	0.957
CNN 2		Time-Frequency	0.999	0.991	0.990	0.988	0.988
CNN 3	Transport Protocol	Time	0.995	0.965	0.968	0.968	0.968
CNN 4		Time-Frequency	0.999	0.998	0.997	0.994	0.991
CNN 5	Tx rate	Time	0.987	0.979	0.978	0.986	0.986
CNN 6		Time-Frequency	0.999	0.999	0.998	0.998	0.998

TABLE 4 Accuracy of all three models, using training, validation, and test datasets, while also depicting precision and recall values for the test dataset, averaged over all classes.

Next, we validate the models using data collected in a real-life setting. Afterwards, we demonstrate the real-life usage of the models through a small-scale prototype.

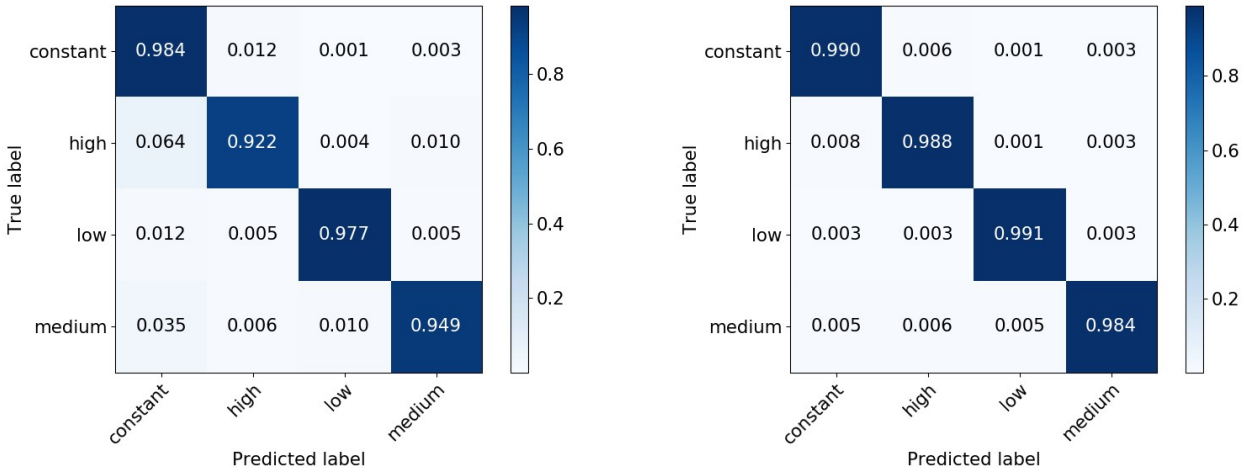
5.1 | Description of generated training datasets

In order to evaluate the designed models, we use the data collection framework presented in Section 4.2 to generate the training, validation, and test datasets used for all experiments presented in Section 5.2. As a topology, we assume one station connected to a Wi-Fi AP at a distance of 5 m. A single flow of traffic was transmitted for 5 s between the two devices with varying rate (100 Kbps, 1 Mbps, 10 Mbps, and 50 Mbps), transport protocol (TCP, or UDP), and transmission pattern of a so-called On-Off application (with 25, 50, 75, 100 % duty cycle). Furthermore, we varied the following Wi-Fi parameters: the standard and frequency (802.11n on 2.4 Ghz, 802.11n on 5 Ghz, 802.11ac on 5 Ghz), the channel width (20 Mhz,40 Mhz), and guard interval (short, long), while assuming the presence of the dynamic Minstrel Rate control algorithm (i.e., dynamic MCS values). Based on these parameters 384 unique experiments were constructed and augmented, leading to over 800 GB of IQ samples. As discussed in the previous section, these samples are processed into two image datasets, time and time-frequency domain, each one composed of 49920 images and requiring around 6 GB of storage. This transformation reduces the storage requirement to less than 1% of the required storage of the raw IQ samples dataset. Finally, each dataset was randomly split to create the training (80%), the validation (10%), and the test (10%) datasets. In the next section, we compare the performance of all constructed models, for the three classification tasks using both datasets.

5.2 | Evaluation using generated synthetic data

Table 4 depicts the accuracy for each model, obtained with respectively, training, validation, and test datasets, for both time (amplitude) and time-frequency (STFT) domains. Additionally, precision and recall are also reported. Overall, it is clear that all models, under all datasets, have an accuracy of above 96 %. Similarly, precision and recall, averaged over all classes per model, are above 0.95. Below we discuss the results for each model individually and show the resulting confusion matrices. These matrices, also known as error matrices, show for each model how all data samples are classified. As such, additional insights into the performance of the models are acquired.

As mentioned in Section 3.2, two models are trained to recognize 4 different traffic patterns (burst versus constant) with a duty cycle of 25 % (low), 50 % (medium), 75 % (high), or 100 % (constant). Table 4 shows the final accuracy of the training, validation, and test datasets after training. CNN 1 denotes the model trained with the dataset in time domain, while CNN 2 is the model that is trained with time-frequency data. Both models, time and time-frequency, achieved above 96 % accuracy in validation and test. However, using time-frequency data leads to higher accuracies above 99 %. This difference, confirmed by other literature as well, is due to the fact that (high amounts of) noise together with fading channel effects have a bigger impact on images that represent the time domain[48]. Note that creating images in time domain is faster, as it requires less complex mathematical computations than STFT. Similar differences are noted for precision and recall. Figure 5 shows the confusion matrices, respectively, for time and time-frequency domain data, obtained from evaluation with the test dataset. According to the confusion matrices shown, the medium and high traffic patterns were the hardest to discriminate, and some of the examples were miss-classified as constant traffic. This is the case for both the time and time-frequency domain. This behavior can be



(a) Using time domain images

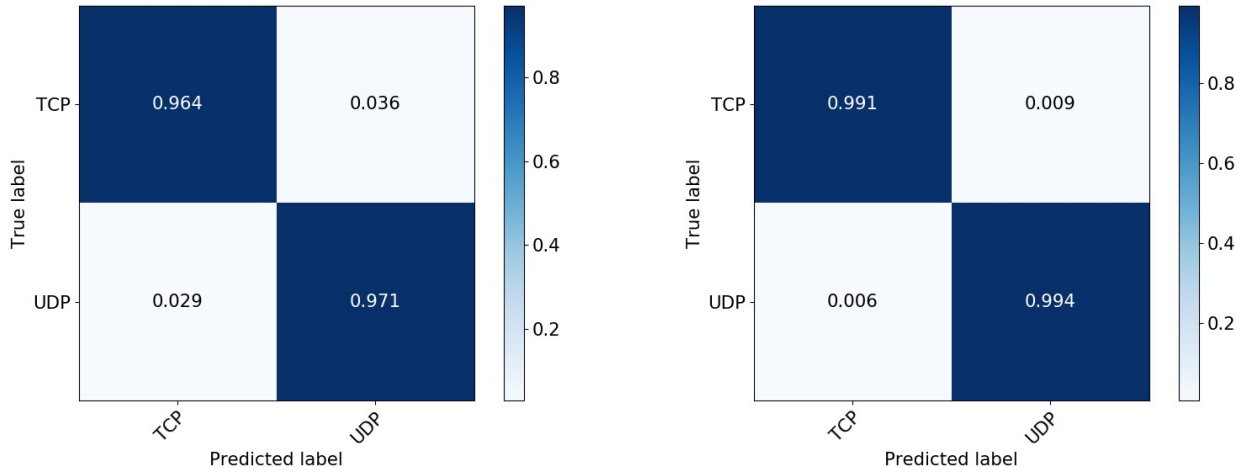
(b) Using time-frequency domain images

FIGURE 5 Normalized confusion matrices for the traffic pattern model

explained due to the MCS adaptation algorithm. If the transmission of a signal is using a lower MCS, then more spectrum will be used in comparison to the same signal using a higher MCS. This in respect to a certain transmission rate and the time window w that is used to create the images, in our case of $w = 0.5$ s.

For the task of discriminating between TCP and UDP traffic, we created two models, one for time domain images and one for time-frequency domain data, to discriminate between two different transport protocols (UDP and TCP). In Table 4 we can see how both models have an accuracy more than 96 % in validation and test. An accuracy of more than 99 % is even possible when using time-frequency images. For time domain, precision and recall are both 0.968, while for time-frequency domain, precision and recall are, respectively, 0.994 and 0.991. This is similar to the trend noticed above for recognition of different traffic patterns. Figure 6 shows the confusion matrices, respectively, for time and time-frequency domain data, obtained from evaluation with the test dataset. These confusion matrices indicate that the miss-classification distribution among the two classes is very similar. This result can be explained by the fact that for scenarios with large traffic flows (i.e., high transmission rates and duty cycles), the spectrum is very occupied and the snapshots and resulting images tend to be more similar for both the TCP and UDP protocols.

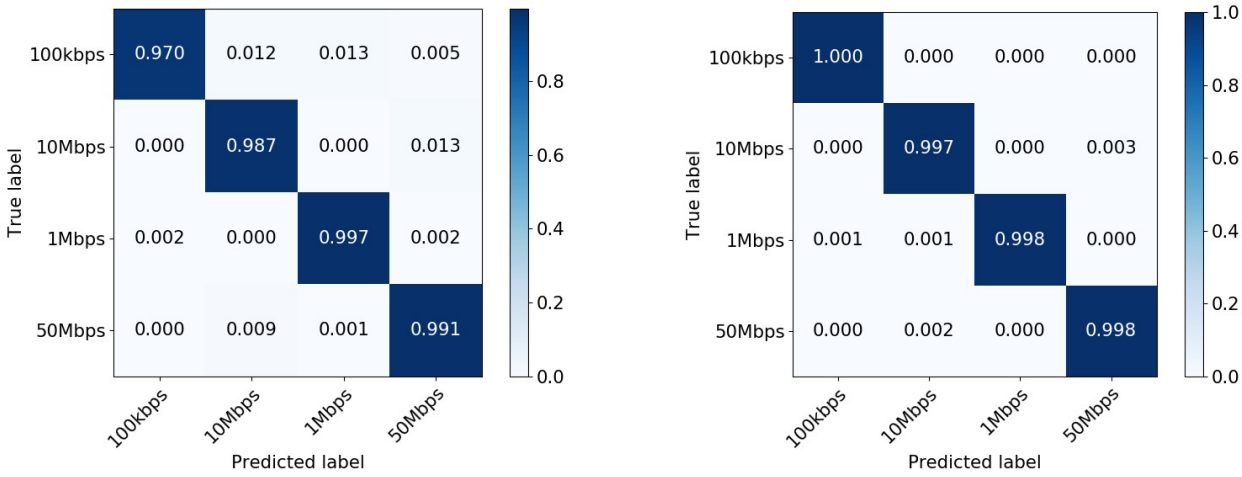
For the third classification task of recognizing different traffic rates, we also designed two models that can recognize the transmission rate of a transmitted traffic flow, based on either time or time-frequency image datasets. The results in Table 4 and Figure 7 show that it is possible to accurately discriminate between different TX rates. In Table 4, CNN 5 denotes the model trained with the dataset in time domain, while CNN 6 is the model that trained with time-frequency data. CNN 5 has an accuracy for the test dataset of 97.8 % and a precision and recall of 0.986. CNN 6 achieves a classification result of 99.8 % and a precision and recall of 0.998. Furthermore, the confusion matrices in Figure 7 show that most of the miss-classification, across both domains, occurred for transmissions of 10 Mbps, which were miss-classified as signals generated at 50 Mbps. When using time domain images, rates of 100 Kbps were also sometimes miss-classified as 1 Mbps or 10 Mbps. Similar to the results of the aforementioned classification tasks, combinations of a high transmission rate with a high duty cycle can lead to samples that look very similar in the spectrum. This is especially true if they are augmented with fading effects and noise, which may be difficult to discriminate by the classifier. However, note these kinds of examples are also responsible for providing a better generalization capacity to the DL models in order to learn and generalize better to unseen data. This is also verified given the high accuracy in both validation and test datasets. Finally, note that the task of classifying traffic patterns and traffic rates can be replaced for more complex regression models in order to predict continues values of these datasets, instead of using classification with predefined labels.



(a) Using time domain images

(b) Using time-frequency domain images

FIGURE 6 Normalized confusion matrices for the traffic protocol model



(a) Using time domain images

(b) Using time-frequency domain images

FIGURE 7 Normalized confusion matrices for the traffic rate model

5.3 | Validation using real-life data

In the previous section, we have used synthetic data, generated through our data generation framework, as test datasets to evaluate the different constructed models. As a next step, we perform an evaluation with data collected in a real-life setting. Besides giving additional insights into the performance of the models, this also presents us with an indication of the applicability of domain randomization within the context of wireless networks. We create a setup consisting of 4 devices: two Intel NUCs and two Software-Defined Radios (SDRs). Each NUC has an i5 core processor, 16 GB of RAM, and an SSD of 500 GB. Each SDR is connected to one of the Intel NUCs, as can be seen in Figure 8. The NUC at the left-hand side is responsible for generating traffic that is transmitted by its connected SDR. The other SDR, at the right side of the figure, captures IQ samples that are stored on the second NUC. There is a distance of 3-4 m between the two SDRs. The captured IQ samples, are afterwards converted into time-domain (amplitude) and time-frequency (STFT) images, using a window size of 5 s. Note that this is the same procedure as for the synthetically generated datasets.

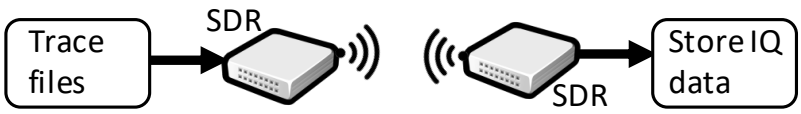


FIGURE 8 Overview of the real-life setup

Model			Accuracy	Precision	Recall
CNN 1	Traffic Pattern	Time	0.541	0.570	0.541
CNN 2		Time-Frequency	0.591	0.593	0.591
CNN 3	Transport Protocol	Time	0.591	0.598	0.591
CNN 4		Time-Frequency	0.782	0.828	0.782
CNN 5	Tx rate	Time	0.734	0.733	0.735
CNN 6		Time-Frequency	0.869	0.876	0.869

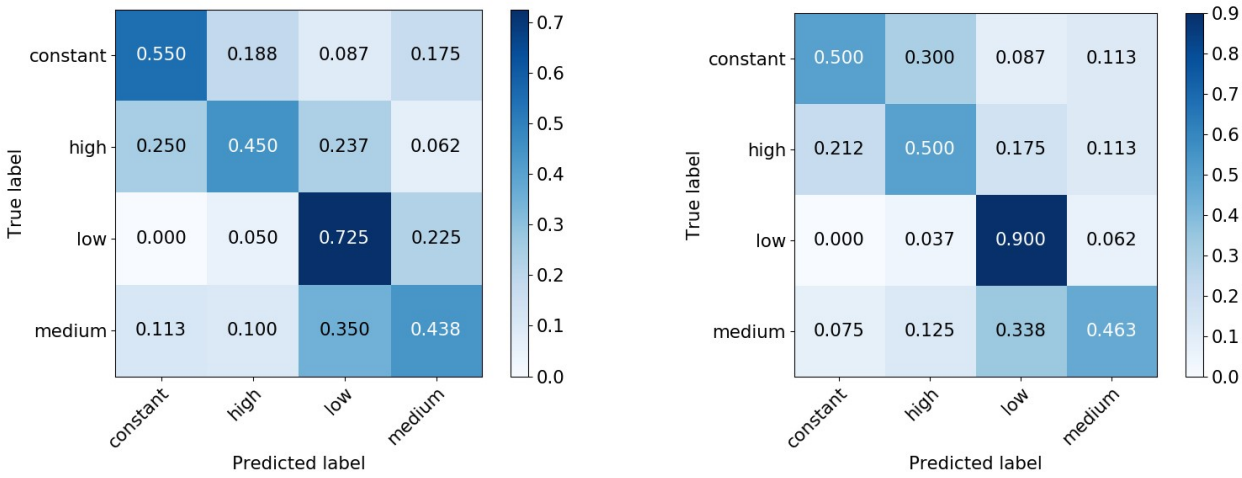
TABLE 5 Accuracy, precision, and recall of all three models using the real-life dataset

For generating the traffic, we make use of the IQ samples that were generated by our data generation framework. We replay those IQ samples using GNU Radio, installed on the NUC connected to the transmitting SDR, that is capable of, among others, transmitting IQ samples[49]. By replaying these samples, we can investigate the robustness of the different modules in a real-life setting, especially the impact of real channel fading and noise effects on the performance. Note that we use the pure IQ samples for this, before the addition of noise and channel fading effects. In particular, we make use of 32 traces that combine all values of the label of our classification models: 4 options for transmission pattern (25 %, 50 %, 75 %, or 100 %)), 2 options for transport protocol (TCP, or UDP), and 4 options for transmission rate (100 Kbps, 1 Mbps, 10 Mbps, or 50 Mbps). Finally, this experiment is conducted in a real-life channel but we checked with a spectrum analyzer for the absence of strong nearby interference sources.

Table 5 depicts the accuracy, precision, and recall for each model, obtained with the real-life dataset, for both time (amplitude) and time-frequency (STFT) domains. Overall, it is clear that there is a significant difference in comparison to the values reported in Table 4. Furthermore, the difference between the results achieved with time domain and time-frequency domain has also grown. We discuss below the results per classification task.

The accuracies for the classification of traffic patterns are shown in Table 5 and the accompanying confusion matrices are presented in Figure 9. The model trained with the dataset in time domain achieves an overall accuracy of 54.1 %, while the model trained with time-frequency images obtains an accuracy of 59.1 %. These values are significantly lower than the ones obtained previously: respectively, 96.2 % and 99.0 %. Similar trends are noticed for the precision and recall. The better accuracy is obtained by using the time-frequency dataset, as motivated previously. From Figure 9, we see that the model mostly miss-classifies the medium, high, and constant traffic patterns. More precisely, medium traffic is roughly in 1 out of 3 cases miss-classified as low traffic, while constant traffic is often miss-classified as high traffic and vice versa. This is the case for both models. Furthermore, note that these miss-classifications already occurred (at much lower scale) with the synthetic test dataset previously. We motivated that behavior by the impact of the MCS adaptation algorithm. Here, we can say that when using real-life data, the errors of the model are amplified.

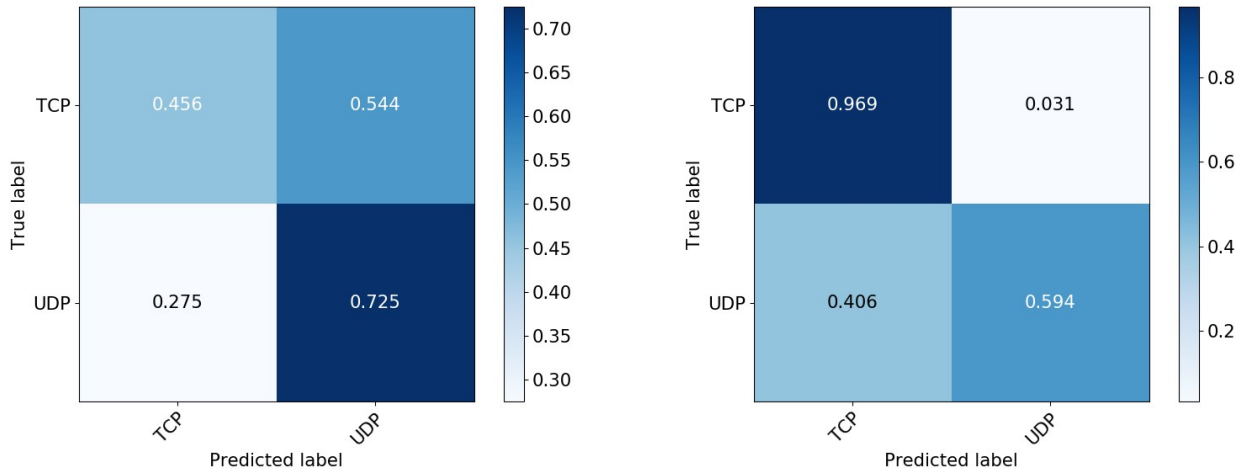
Correctly discriminating TCP and UDP is successfully done in 59.1 % of the time for the model in time domain, and 78.2 % of the time for the time-frequency model. While both accuracies are still lower than reported previously with the test dataset (respectively, 96.8 % and 99.7 %), the difference between the two models is very large. The same difference can be seen for precision and recall. The difference in accuracy for CNN 4 (time-frequency) using test or real-life data is roughly only half the size of the difference recorded for CNN 3 (time). When considering the confusion matrices in Figure 10, the difference between the two models is even more remarkable. The time domain model can better classify UDP traffic, with an accuracy of 72.5 %, while the time-frequency model can classify TCP traffic with an accuracy of 96.9 %. The latter value is relatively close the accuracy obtained with the test data (99.1 %), as reported in Figure 6b. We believe that the reason for these results is two-fold. First, as mentioned previously, the spectrum looks rather similar for different scenarios with high transmission rates and duty cycles, making it harder to distinguish between TCP and UDP protocols. Second, the time-domain models and images are



(a) Using time domain images

(b) Using time-frequency domain images

FIGURE 9 Normalized confusion matrices for the traffic pattern model



(a) Using time domain images

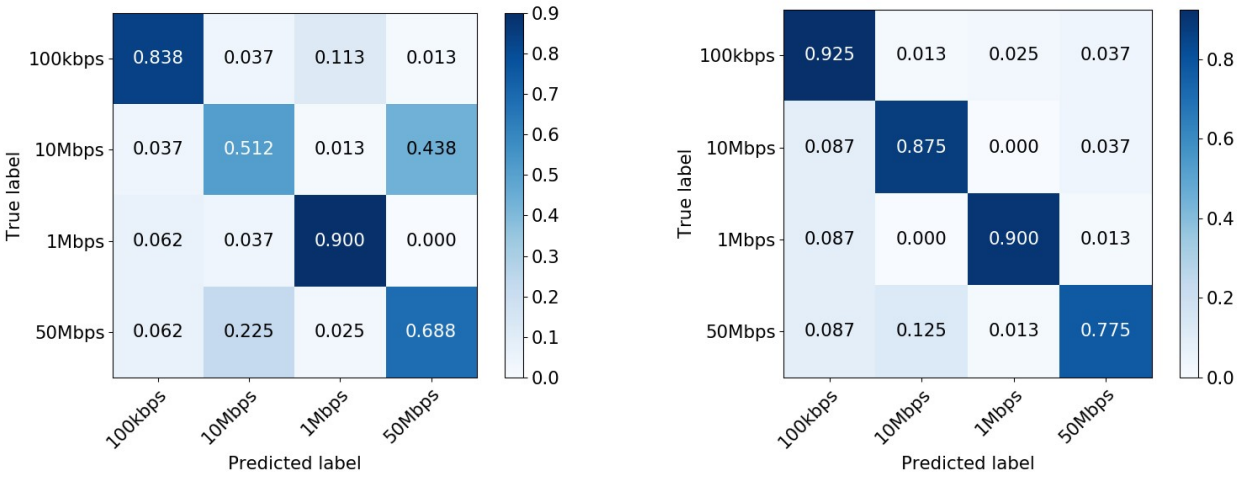
(b) Using time-frequency domain images

FIGURE 10 Normalized confusion matrices for the traffic protocol model

more susceptible to channel effects and noise. This could make TCP ACK transmissions harder to detect, as such explaining the significant lower accuracy of CNN 3 for TCP classification.

The third classification task of detecting different transmissions rates clearly achieves the best overall accuracies. As reported in Table 5, we see that CNN 5 (time) and CNN 6 (time-frequency) achieve, respectively, an accuracy of 73.4 % and 86.9 %. This in contrast to the values reported earlier for the test dataset: 97.8 % and 99.8 %, respectively. For precision and recall, the same observation is made. From the confusion matrices, shown in Figure 11, we can conclude that the number of miss-classifications increases when increasing the transmission rate. Across both models, the labels of 10 Mbps and 50 Mbps are the hardest to classify. This can be explained by the strong similarity in the spectrum between both cases, as mentioned above.

Overall, we can conclude that the different models can cope relatively well with the unseen real-life data samples. This is especially the case for the classification of transport protocol and transmission rates, using time-frequency data (i.e., CNN 4 and CNN 6). However, we also acknowledge that the presented models should be further improved to boost overall accuracy. The



(a) Using time domain images

(b) Using time-frequency domain images

FIGURE 11 Normalized confusion matrices for the traffic rate model

latter is especially true for the classification of traffic patterns based on the duty cycle. When comparing the confusion matrices obtained from both synthetic and real-life validation, we notice that the already present miss-classifications of certain labels are enlarged upon using real-life data. We can conclude that the ideas behind domain randomization can indeed be applied to the recognition of patterns within wireless contexts and can help in making the models more robust. Future work should consider enhancing the models, in order to reduce the miss-classifications for both the synthetic and real-life datasets. Furthermore, we can also consider constructing regression models for the recognition of different traffic rates and patterns, the recognition of multiple flows, or the use of semi-supervised learning models to better cope with unseen data. We will discuss this further in Section 6.

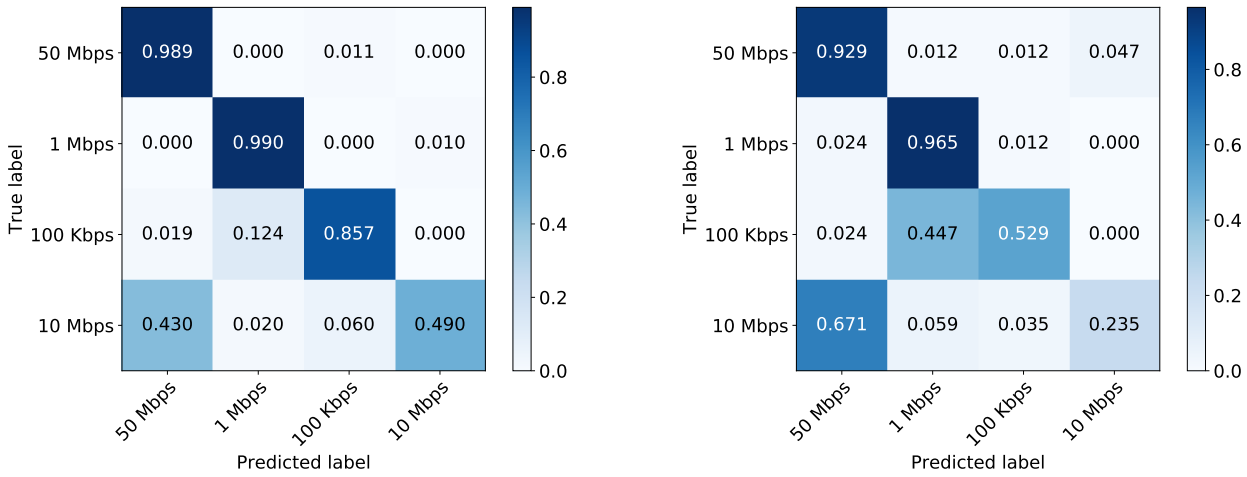
5.4 | Prototype demonstrator

So far, the performance of our models has been evaluated on infrastructures and servers with considerable amounts of resources. However, we will now demonstrate how our models can be used in a real-life setting. For this, we make use of the setup created for the real-life data collection in Section 5.3 and illustrated in Figure 8. The used hardware and the connections between the different devices are identical. However, instead of using the Intel NUC at the receiving side (at the right side of Figure 8) only for storage, we will now also implement the data adaptation steps and a classification model. As such, demonstrating the real-life applicability on more every-day devices. For this prototype, we implement the classification model for transmission rate, using time domain images (denote as CNN 5 before). Time domain images are chosen because they require less complex mathematical computations than STFT, making them faster to generate, which is important in a real-time setting. For the traffic flows, we select the generated IQ samples of four different scenarios with different characteristics. We selected traces with different values for the transmission rate, while other label values were chosen arbitrarily. As before, these IQ samples are transmitted using GNU Radio. The selected traffic classes and their characteristics are depicted in Table 6. The experiment was conducted in both an interference-free and an occupied channel in the 2.4 GHz frequency band.

The resulting confusion matrices for both runs can be seen in Figure 12. In total 250 snapshots of 5 s per traffic stream were created for each experiment. The overall accuracy for the first experiment in the empty channel is 83.0 %. For the second experiment in the occupied channel, an accuracy of 66.5 % is achieved. From both confusion matrices, it is clear that the most miss-classifications occurred for the 10 Mbps traffic (i.e., trace 3). This is similar to the results presented in the previous sections, where also 10 Mbps traffic was regularly classified as 50 Mbps traffic. Furthermore, we noticed that it took up to 5-6 s to convert the captured IQ samples to the images in time domain. This was especially the case for the 50 Mbps traffic, while the smaller 100 Kbps traffic took less than half of that time. This data adaption time can be decreased by implementing this feature directly in hardware or averaging IQ samples directly on the capturing interface, before converting them to images.

Traffic class	Pattern	Protocol	Rate
Trace 0	High (75 %)	UDP	50 Mbps
Trace 1	Medium (50 %)	TCP	1 Mbps
Trace 2	Low (25 %)	UDP	100 Kbps
Trace 3	Medium (50 %)	TCP	10 Mbps

TABLE 6 Traffic classes and their characteristics for prototype



(a) interference-free channel

(b) Occupied channel

FIGURE 12 Normalized confusion matrices for the prototype

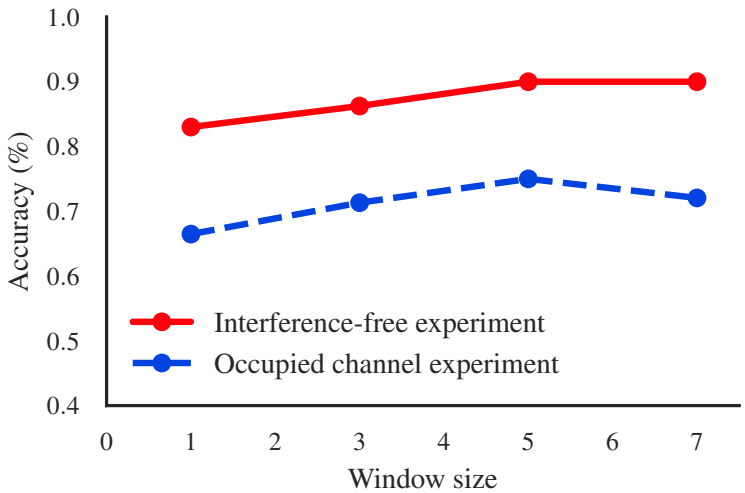
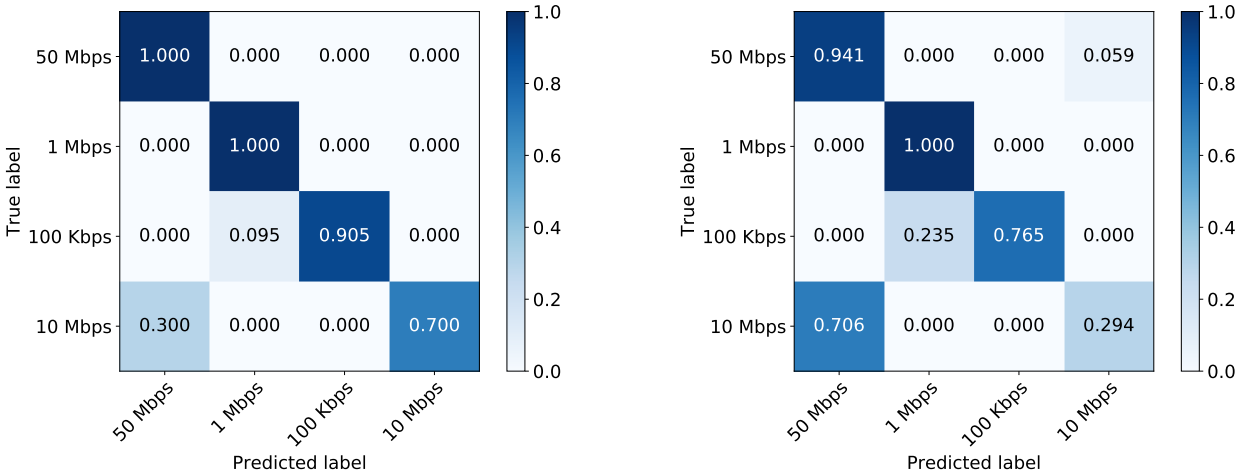


FIGURE 13 Overview of accuracies obtaining using difference window sizes

Finally, so far we have always reported the accuracy for all the models by using a single classification result. However, an application using the reported results could take the average result over a number of snapshots. This can, for instance, be done using a sliding window. We have tested this out for window sizes of 3, 5, and 7. Figure 13 shows the resulting accuracies across



(a) interference-free channel

(b) Occupied channel

FIGURE 14 Normalized confusion matrices for the prototype using a combination of 5 snapshots

different window sizes for both experiments The best results are obtained when using an average over 5 samples, per traffic class: 90.0 % and 75.0 %, respectively for the experiments with an interference-free channel and an occupied channel. This means that there is an increase of, respectively, 7 % and 8.5 %. The resulting confusion matrices for this optimal window size are shown in Figure 14.

6 | CONCLUSION AND FUTURE WORK

In this paper, we have presented a traffic classification approach at the spectrum level. This approach can, among others, be used to provide monitoring information to intelligent spectrum management solutions. In contrast, to existing DPI and packet-based methods, the listening device does not have to be part of the network and respects modern data privacy regulations. In particular, we present a CNN architecture that forms the basis for prediction models to recognize TCP and UDP traffic, burst traffic with different duty cycles, and different transmission rates. Furthermore, we have presented a data generation framework by combining two state-of-the-art simulators. As such, we are the first to explore principles behind successful domain randomization techniques in the context of wireless networks. Using our data generation framework, we created two different datasets containing images that represent snapshots of the spectrum in either time or time-frequency domain. Our evaluation shows that both approaches have an accuracy of more than 96 % for all models. However, the models trained with the dataset with images in the time-frequency domain, obtained after performing STFT, offer a higher accuracy of more than 99 %. When validating with real-life data, the accuracy of the models dropped, as expected. However, it is still possible to recognize different rates with an accuracy of around 87 %. This means that domain randomization can indeed be a powerful tool to use within the context of wireless networks. Future work will focus on improving the models by, for instance, using semi-supervised or transfer learning methods that can better cope with unseen data. New models should also be developed that can both discriminate novel features (e.g., the detection of multiple rates through regression or the recognition of application-level information) and can cope with more complex settings (e.g., multiple flows, interference of other networks, or multiple transmitters). Combining sequences of short time windows of the spectrum data with LSTM or combined CNN- LSTM architectures need to be further explored in order to see if complex traffic patterns can be classified with higher accuracy in more complex scenarios such as multiple transmitters with multiple flows and higher transmission rates, e.g., via image segmentation strategies. Furthermore, the full power of domain randomization should also be explored.

References

1. Qazi ZA, Lee J, Jin T, Bellala G, Arndt M, Noubir G. Application-awareness in SDN. *ACM SIGCOMM computer communication review* 2013; 43(4): 487–488.

2. AbuHmed T, Mohaisen A, Nyang D. A survey on deep packet inspection for intrusion detection systems. *arXiv preprint arXiv:0803.0037* 2008.

3. Pacheco F, Exposito E, Gineste M, Baudoin C, Aguilar J. Towards the deployment of Machine Learning solutions in network traffic classification: A systematic survey. *IEEE Communications Surveys & Tutorials* 2018.

4. Cisco . Cisco Visual Networking Index: Forecast and Trends, 2017âĂŞ2022. 2017. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf. Accessed January 29, 2020.

5. Andrews JG, Buzzi S, Choi W, et al. What will 5G be?. *IEEE Journal on selected areas in communications* 2014; 32(6): 1065–1082.

6. Sharma SK, Bogale TE, Chatzinotas S, Ottersten B, Le LB, Wang X. Cognitive radio techniques under practical imperfections: A survey. *IEEE communications surveys and tutorials* 2015.

7. Abinader FM, Almeida EP, Chaves FS, et al. Enabling the coexistence of LTE and Wi-Fi in unlicensed bands. *IEEE Communications Magazine* 2014; 52(11): 54–61.

8. Lotfollahi M, Zade RSH, Siavoshani MJ, Saberian M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *arXiv preprint arXiv:1709.02656* 2017.

9. Peng XB, Andrychowicz M, Zaremba W, Abbeel P. Sim-to-real transfer of robotic control with dynamics randomization. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE. ; 2018: 1–8.

10. Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P. Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. ; 2017: 23–30.

11. Tremblay J, Prakash A, Acuna D, et al. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. ; 2018: 969–977.

12. Sherry J, Lan C, Popa RA, Ratnasamy S. Blindbox: Deep packet inspection over encrypted traffic. *ACM SIGCOMM Computer Communication Review* 2015; 45(4): 213–226.

13. Nguyen TT, Armitage GJ. A survey of techniques for internet traffic classification using machine learning.. *IEEE Communications Surveys and Tutorials* 2008; 10(1-4): 56–76.

14. Zhang J, Chen X, Xiang Y, Zhou W, Wu J. Robust network traffic classification. *IEEE/ACM Transactions on Networking (TON)* 2015; 23(4): 1257–1270.

15. Wang W, Zhu M, Zeng X, Ye X, Sheng Y. Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN). IEEE. ; 2017: 712–717.

16. Shi M, Laufer A, Bar-Ness Y, Su W. Fourth order cumulants in distinguishing single carrier from OFDM signals. In: MILCOM 2008-2008 IEEE Military Communications Conference. IEEE. ; 2008: 1–6.

17. Karami E, Dobre OA. Identification of SM-OFDM and AL-OFDM signals based on their second-order cyclostationarity. *IEEE transactions on vehicular technology* 2015; 64(3): 942–953.

18. Karami E, Dobre OA, Adnani N. Identification of GSM and LTE signals using their second-order cyclostationarity. In: 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. IEEE. ; 2015: 1108–1112.

19. Liu W, Kulin M, Kazaz T, Shahid A, Moerman I, De Poorter E. Wireless technology recognition based on RSSI distribution at sub-Nyquist sampling rate for constrained devices. *Sensors* 2017; 17(9): 2081.

20. Schmidt M, Block D, Meier U. Wireless interference identification with convolutional neural networks. In: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE. ; 2017: 180–185.

21. Jeong S, Lee U, Kim SC. Spectrogram-Based Automatic Modulation Recognition Using Convolutional Neural Network. In: 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE. ; 2018: 843–845.

22. Rajendran S, Meert W, Giustiniano D, Lenders V, Pollin S. Deep learning models for wireless signal classification with distributed low-cost spectrum sensors. *IEEE Transactions on Cognitive Communications and Networking* 2018; 4(3): 433–445.

23. O'Shea TJ, Corgan J, Clancy TC. Convolutional radio modulation recognition networks. In: International conference on engineering applications of neural networks. Springer. ; 2016: 213–226.

24. Kulin M, Kazaz T, Moerman I, De Poorter E. End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications. *IEEE Access* 2018; 6: 18484–18501.

25. Yi S, Wang H, Xue W, et al. Interference Source Identification for IEEE 802.15. 4 wireless Sensor Networks Using Deep Learning. In: 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). IEEE. ; 2018: 1–7.

26. Testi E, Favarelli E, Giorgetti A. Machine Learning for User Traffic Classification in Wireless Systems. In: 2018 26th European Signal Processing Conference (EUSIPCO). IEEE. ; 2018: 2040–2044.

27. O'Shea TJ, Hitefield SD, Corgan J. End-to-End Radio Traffic Sequence Recognition with Deep Recurrent Neural Networks. *arXiv* 2016; http://arxiv.org/abs/1610.00564.

28. Bai S, Kolter JZ, Koltun V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* 2018; http://arxiv.org/abs/1803.01271.

29. Goodfellow I, Bengio Y, Courville A. *Deep learning*. MIT press . 2016.

30. Tschopp F, Martel JN, Turaga SC, Cook M, Funke J. Efficient convolutional neural networks for pixelwise classification on heterogeneous hardware systems. In: 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI). IEEE. ; 2016: 1225–1228.

31. Chen Y, Wang J, Zhu B, Tang M, Lu H. Pixel-wise deep sequence learning for moving object detection. *IEEE Transactions on Circuits and Systems for Video Technology* 2017.

32. Guo R, Liu J, Li N, et al. Pixel-Wise Classification Method for High Resolution Remote Sensing Imagery Using Deep Neural Networks. *ISPRS International Journal of Geo-Information* 2018; 7(3): 110.

33. LeCun Y, Bengio Y. Convolutional Networks for Images, Speech, and Time Series. In: The Handbook of Brain Theory and Neural Networks. MIT Press. 1998 (pp. 255–258).

34. Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience* 2018; 2018.

35. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. ; 2011: 315–323.

36. Dahl GE, Sainath TN, Hinton GE. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE. ; 2013: 8609–8613.

37. O'Shea K, Nash R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* 2015.

38. Kingma DP, Ba J. Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR). ; 2015.

39. Chollet F, others . Keras. https://keras.io. Accessed January 29, 2020.

40. Abadi M, others . TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Software available from tensorflow.org.

41. Tobin J, Biewald L, Duan R, et al. Domain randomization and generative models for robotic grasping. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. ; 2018: 3482–3489.

42. Henderson TR, Lacage M, Riley GF, Dowell C, Kopena J. Network simulations with the ns-3 simulator. *SIGCOMM demonstration* 2008; 14(14): 527.

43. Mathworks Products and Services. https://www.mathworks.com/products.html. Accessed January 29, 2020.

44. Zheleva M, Chandra R, Chowdhery A, Kapoor A, Garnett P. TxMiner: Identifying transmitters in real-world spectrum measurements. In: 2015 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN). ; 2015: 94-105

45. Zheleva M, Bogdanov P, Larock T, Schmitt P. AirVIEW: Unsupervised transmitter detection for next generation spectrum sensing. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. ; 2018: 1673-1681

46. Camelo M, Shahid A, Fontaine J, et al. A Semi-Supervised Learning Approach Towards Automatic Wireless Technology Recognition. In: ; 2019: 11–14.

47. Shahid A, Fontaine J, Camelo M, et al. A Convolutional Neural Network Approach for Classification of LPWAN Technologies: Sigfox, LoRA and IEEE 802.15.4g. In: 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). ; 2019: 1-8

48. Kulin M, Kazaz T, Moerman I, De Poorter E. End-to-End Learning From Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications. *IEEE Access* 2018; 6: 18484-18501.

49. GNU Radio: The free & Open Software Radio Ecosystem. https://www.gnuradio.org/. Accessed January 29, 2020.