A General Approach for Traffic Classification in Wireless Networks using Deep Learning

Miguel Camelo, Paola Soto, and Steven Latré,

***Abstract*—Traffic Classification (TC) systems allow inferring the application that is generating the traffic being analyzed. State-of-the-art TC algorithms are based on Deep Learning (DL) and have outperformed traditional methods in complex and modern scenarios, even if traffic is encrypted. Most of the works on TC assume the traffic flows on a wired network under the same network management domain. This assumption limits the capabilities of TC systems in wireless networks since users' traffic on one network domain can be negatively impacted by undetected traffic transmissions from users in other network domains or detected ones but with no traffic context in a shared spectrum. To solve this problem, we introduce a novel framework to achieve TC at any layer on the radio network stack. We propose a spectrum-based procedure that uses a DL-based classifier to realize this framework. We design two DL-based classifiers, a novel Convolutional Neural Network (CNN) spectrum-based TC and a Recurrent Neural Networks (RNN) as baseline architecture, and benchmark their performance on three TC tasks at different radio stack layers. The datasets were generated by combining packet traces from real transmissions with a standard-compliant waveform generator for 802.11 radio technologies. Performance evaluations show that the best model can achieve an accuracy above 92% in the most demanding TC task, a drop of only 4.37% in accuracy compared to a byte-based DL approach, with micro-second per-packet prediction time, which is very promising for delivering real-time spectrum-based traffic analyzers.**

***Index Terms*—Deep Learning, Intelligent Radios, Radio Spectrum, Traffic Classification, Wireless Networks**

I. Introduction

Nowadays, wireless technologies are omnipresent and provide access to millions of users and machines to the Internet. This access is mostly offered by complementary technologies, e.g., 4G/5G Mobile Networks and Wireless Local Area Networks (WLANs), with an increased network capacity to support the ever-increasing number of devices and applications. As a result, managing and optimizing the wireless network capacity to provide Quality of Service (QoS) becomes even more challenging [1], [2]. Traditionally, the Network Monitoring Service (NMS) performs a set of tasks to analyze the behavior of the networks and services throughout their traffic. The information provided by this system can be used to determine which applications affect network the most in terms of total bandwidth usage or identify the most critical links so decision-making engines for network management can ensure fast troubleshooting and securing high QoS to the users.

More recently, this service has been enhanced with Machine Learning (ML) techniques to perform automatic network traffic analysis such as network state predictions, anomaly detection, malware detection, and TC [3]. Focusing on TC, this network management task allows inferring the application that is generating the traffic [2], [3]. Knowing the traffic class provides a mechanism to enforce specific security and QoS policies on the analyzed traffic. Over the years, several approaches have been designed and developed to follow the evolution of the technologies driving the development of user's application and communication protocols. More recently, DL-based traffic classifiers have outperformed traditional methods such as port-based classifiers, Deep Packet Inspection (DPI), and flow-based traffic analysis using statistical ML in complex and modern scenarios even where the traffic is encrypted [4]–[6].

In general, the TC task is assumed to be performed on traffic that belongs to the same network domain and over a byte/protocol representation of the packet at the Link Layer (L2) (or above). These assumptions limit the capabilities of TC systems in wireless networks using shared spectrum, e.g., in unlicensed bands. The users' traffic from one wireless network domain can be negatively impacted by users' traffic transmissions from other wireless networks without being noticed by the TC system as demonstrated in [7]. Contrary to a wired network, co-located wireless transmissions in the same spectrum band can generate Physical Layer (L1) packets that are not be detected by a receiver performing the TC task. Examples of these cases are when the transmitter is using a wireless technology that cannot be demodulated and decoded by the receiver (i.e., different technology) or when the transmission can be demodulated and decoded (i.e., same technology) but the decoded traffic is already encrypted in L2 (e.g., wireless devices belong to a different wireless network domain and its network is secured).

To overcome this limitation, we need to move from TC systems that work at a byte representation of the packet to TC systems at spectrum level. In this way, the traffic generated by any other wireless device sharing the same spectrum can be monitored, detected, assembled and classified, even if it is encrypted, belongs to a different network domain or uses various wireless technologies. However, designing and deploying TC algorithms that work on a spectrum representation of the packets (or flow of packets) adds new challenges that are not present when using the byte/protocol representation of it: spectrum-based packets are modulated, coded, and some times encrypted before being transmitted. As a result, transmitting the same user's L2 packet may result in a very different spectrum view of the packet using either the same wireless technology, e.g., due to different Modulation and Coding

M. Camelo, P. Soto, and S. Latré are with University of Antwerp - imec, IDLab, Department of Mathematics and Computer Science, Sint-Pietersvliet 7, 2000 Antwerp, Belgium.

Acronym	Description
AE	AutoEncoder
AI	Artificial Intelligence
AP	Access Point
ATSSS	3GPP Access Traffic Steering Switching and Splitting
BPSK	Binary Phase Shift Keying
CCK	Complementary Code Keying
CIR	Collaborative Intelligent Radio
CIRN	Collaborative Intelligent Radio Networks
CNN	Convolutional Neural Network
Conv	Convolutional
CR	Cognitive Radio
CU	Centralized Unit
CWT	Continuous Wavelet Transform
DCI	Downlink Control Information
DL	Deep Learning
DNN	Deep Neural Network
DPI	Deep Packet Inspection
DSSS	Direct-Sequence Spread Spectrum
DT	Decision Tree
DU	Distributed Unit
FFT	Fast Fourier Transform
GB	Gradient Boost
GL	Gossip Learning
GP	Gaussian Processes
GRU	Gated Recurrent Units
GW	Gateway
HDLC	High-level Data Link Control
ICDE	Intelligent Control and Decision Engine
ImRAT	Intelligent multi-RAT
IQ	In-phase and Quadrature components
K-NN	k-Nearest Neighbours
L1	Physical Layer
L2	Link Layer
L7	Application Layer
LR	Logistic Regressor
LSTM	Long Short-Term Memory
LTE	Long-Term Evolution
LTE-PDCCH	LTE Physical Downlink Control CHannel
MCS	Modulation and Coding Scheme
ML	Machine Learning
MLP	Multi-Layer Perceptron
MTL	Multi-Task Learning
NB	Naïve Bayes
NMS	Network Monitoring Service
NN	Neural Network
OFDM	Orthogonal Frequency Division Multiplexing
PHY	Physical-layer
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAN	Radio Access Network
RAT	Radio Access Technologies
ReLU	Rectified Linear Unit
RF	Random Forest
RIC	RAN Intelligent Controller
RNN	Recurrent Neural Networks
RRU	Remote Radio Unit
SC2	Gaussian Processes
SDAE	Stacked Denoising AutoEncoder
SDR	Software Defined Radios
SNR	Signal-To-Noise-Ratio
STFT	Short-Time Fourier Transform
SVM	Supported Vector Machine
TC	Traffic Classification
TR	Technology Recognition
UT	User's Terminal
VAE	Variational AutoEncoder
WLAN	Wireless Local Area Network
WNIC	Wireless Network Interface Card
WPA	Wi-Fi Protected Access

TABLE I: List of acronyms used in this paper

Schemes (MCSs), or a different one, e.g., due to different digital multi-carrier transmission schemes.

To this extend, we introduce a novel framework to achieve TC at any layer on the radio network stack. Building on top of it, a procedure based on DL to perform TC on spectrum samples is proposed. This procedure enables the management algorithms running at the Gateway (GW) nodes (or beyond) to perform better by having a broader view of the traffic flowing in the shared spectrum. The main contributions of this paper are summarized as follows:

- We present a general framework that enables the development of TC algorithms optimized for wireless networks. Up to the best of our knowledge, this is the first framework that allows developing Radio Access Technologies (RAT)-agnostic spectrum-based TC algorithms.
- We proposed a spectrum-based TC procedure that exploits the proposed framework's functional blocks and works on L1 packets. Compared to similar works like [8], the proposed procedure includes not only the traffic classifier designing but also the complete chain to achieve spectrum-based TC. Moreover, the proposed approach removes the need for specialized algorithms to separate/aggregate users' traffic flows (e.g., a radio identification procedure [9]), as the one required on recent approaches like [10] , as it uses as classification object single L1 packets (as a sequence of raw IQ samples).
- We designed and evaluated a DL architecture based on CNNs to solve the task of classifying packets directly on spectrum data. Up to the best of the authors knowledge, this is the first work that uses this type of Deep Neural Network (DNN) to solve such task. Moreover, we demonstrate that the proposed architecture outperform a RNN architecture, also used in [8] and that is traditionally used to solve classification problems with time series as input data,
- We present the first detailed analysis on the performance achieved by different spectrum-based classifiers using DL architectures in terms of classification accuracy on different classification tasks at different radio stack layers (L2 and L7), including a comparison against encrypted L2 byte-based packet classifiers, and execution time in training and inference using 802.11 standard-compliant L1 packets and with input sequences of more than 3K spectrum samples. Moreover, these results provide initial insights about the feasibility of this approach for real-time classification. These evaluations complement and extend previous results where the performance of DL approaches solving the TC problem have been evaluated and compared using byte [11] and spectrum representation of the packets [8], [10].
- We create and provide an open source dataset that contains 802.11 standard-compliant L1 waveforms. The waveforms are generated by different 802.11 technologies (b, g, n), which results in different transmission schemes such as Direct-Sequence Spread Spectrum (DSSS) in 802.11b and Orthogonal Frequency Division Multiplexing (OFDM) in 802.11g/n, different types of L2 frames

(management, control and data), and multiple MCS (modulations such as Binary Phase Shift Keying (BPSK) and Complementary Code Keying (CCK) for 802.11b and BPSK and Quadrature Phase Shift Keying (QPSK), 16-Quadrature Amplitude Modulation (QAM), 64-QAM for 802.11g/n with coding rates of 1/2, 3/4, and 5/6 according to the standard and modulation selected). Moreover, the payload carried by this L1 packets (information at L2 and above) were generated using real traces of L7 application running on a mobile device and connected to a secured 802.11 Access Point (AP) with Wi-Fi Protected Access (WPA)-2. This is the first open and available dataset for testing traffic classification at spectrum level and we believe this dataset would foster reproducibility and allow further advances on this topic. The dataset and the code associate to this paper can be obtained in Zenodo[1][2] and Github[3].

The remainder of this paper is structured as follows. Table I lists the acronyms used in this paper. We present the related works in Section II. The general framework for TC is introduced in Section III, and the proposed spectrum-based TC algorithm using DL is presented in Section IV. Finally, we show the DL models' performance evaluation results on both coarse-grained and fine-grained traffic classification tasks in Section VI and conclusions and future work in Section VII.

II. Related work

This section presents some of the most relevant work on TC using DL approaches for encrypted traffic. For a more exhaustive literature review on the general applications of DL in wireless networks and on ML/DL approaches for TC, we refer the reader to [5] and [4], [12], respectively.

A. TC using L2 (and above) classification objects

Over the years, the applications have evolved, and so the algorithms and techniques used to classify the traffic generated by them [2]–[4], [13], [14]. The initial approach was using port numbers. Later, traffic classifiers using DPI techniques were designed to find patterns in the data packets' payload. While port-based classifiers are more accessible and faster than DPI methods, DPI outperformed them at the cost of higher computational requirements. Unfortunately, both approaches are limited to non-encrypted traffic that typically belongs to the same network domain. To avoid this problem, ML approaches were proposed to classify the traffic using packet flows, where flow-measurements are used as features. However, it has been shown that its accuracy can be affected by variations of the user's behavior, device OS-specific patterns, network-specific conditions, among others [14].

More recently, TC algorithms based on DL approaches have outperformed ML ones based on traditional algorithms [3], [4], [12]. One example of these algorithms is shown in [15] and [16], where authors proposed an end-to-end TC algorithm based on CNNs that converts raw traffic into images. In [15], the authors took the first several packets of the traffic flows and its time-series features and were able to identify the application or protocol type that generates them. The Seq2Img model was compared against four popular classifiers such as Supported Vector Machine (SVM), Multi-Layer Perceptron (MLP), Naïve Bayes (NB), and Decision Tree (DT). Results showed that all approaches perform equally well when classifying protocols, but Seq2Img is almost 12% more accurate than other models when classifying applications.

To overcome the problem of data labeling, authors in [17] proposed a semi-supervised approach that pre-trains a 1D-CNN model on an unlabeled dataset to infer traffic patterns and afterward re-train the model on a labeled dataset to confirm those patterns. In this way, the amount of labeled data needed in the second step is considerably reduced. The datasets are based on time-series features of a fixed number of sampled packets from traffic flows. Results showed that the pre-training step increases the model's accuracy by up to 10% compared to a model without pre-training. A Stacked Denoising AutoEncoder (SDAE), a CNN, and a Long Short-Term Memory (LSTM) were proposed as classifiers in [18], where Netlog was developed to simplify the data labeling process. Comparison against a Random Forest (RF), which requires statistical features, and an MLP showed that all DL models performed over 20% better in terms of accuracy than RF, showing that much more insightful features can be learned from raw data than the statistical features used by RF.

A combination of two CNN layers followed by one LSTM layer with two fully connected layers at the end was proposed in [19]. The time-series features are taken from the headers of the first 20 packets exchanged during the flow lifetime and did not include any information that could identify users (MAC/IP addresses) to ensure data confidentiality. Results showed that the combination obtained the best results in terms of accuracy and F1-score. AutoEncoders (AEs) are also a predominant option as models. For instance, in [20], a network traffic flow is also transformed into an image that is later processed by a semi-supervised model based on a Variational AutoEncoder (VAE). The authors took data from the HTTP sessions (requests and responses) and converted them to a 28x36 image. The proposed VAE uses an MLP encoder and decoder that analyses images in an unsupervised manner as feature extractor. Then, the extracted features are mapped to an app in a supervised manner. Results show that, even with two features, the network traffic can be effectively discriminated in the unsupervised step achieving an accurate classification at the supervised step.

As pointed out by the authors in [21], most of the literature in the field of NMS are focused on single-task learning, e.g., each model is designed and trained to solve one specific learning task such as TC, traffic prediction, or anomaly detection. As a solution, Multi-Task Learning (MTL) approaches have been proposed in [21] and [22], where TC is used as one of the learning tasks, to leverage useful information contained in multiple related tasks aiming to improve the generalization capabilities of all them while learning. The authors in [21] used a MTL approach to jointly solve the TC and traffic

[1]https://doi.org/10.5281/zenodo.5208201
[2]https://www.doi.org/10.5281/zenodo.5208627
[3]https://github.com/miguelhdo/tc_spectrum

TABLE II: Comparison of our work and other contributions focusing on TC using DL

Contributions	Traffic Representation	Classification at any layer	Classification object as input data	End-to-End framework	Proposed Model	Comparison against other models	Comparison against other traffic representation	Multi-task learning	Dataset and availability
[15]	L2 Packet flow	No	Flow statistics as images	Yes	2D-CNN	SVM, MLP, NB, DT	No	No	Private
[16]	L2 Packet flow	No	Session and Flow statistics	Yes	1D-CNN	2D-CNN	No	No	Public
[17]	L2 Packet flow	No	24 statistical features of a packet	Yes	1D-CNN	1D-CNN trained in supervised mode	No	No	QUIC, Public
[18]	L3/L4 packet	No	Raw bytes	Yes	SDAE, CNN, LSTM-RNN	RF, MLP	No	No	Restricted
[19]	L2 Packet flow	No	Sequence of packet statistics	Yes	CNN+LSTM-RNN	CNN, LSTM-RNN	No	No	RedIRIS, Public
[20]	L7 packet	No	Raw bytes as images	Yes	VAE	No	No	No	IMT17, Public
[21]	Raw DCI	No	Transport Block Size	Yes	AE + softmax +dense layer.	LSTM-RNN+softmax, LSTM-RNN+dense layer	No	Yes	Private
[23]	Raw DCI	No	Transport Block Size	Yes	LSTM-RNN, 1D-CNN, MLP	SVM, LR, K-NN, RF, GP	No	No	Private
[8]	L1 packet	Yes	Raw IQ samples	No	LSTM-RNN	No	No	No	Private
[10,24]	L1 packet flow	Yes	Raw IQ samples as images	No	2D-CNN	2D-CNN on raw L2 packets	Yes, raw L2 packet	No	Private
This work	**L1 packet**	**Yes**	**Raw IQ samples**	**Yes**	**2D-CNN**	**GRU-RNN, GB**	**Yes, raw L2 packet and packet length**	**No**	**IDLAB-TC-SPECT, Public**

prediction task using traffic traces containing the Downlink Control Information (DCI) messages carried within the LTE Physical Downlink Control CHannel (LTE-PDCCH) with a time granularity of 1ms. Their proposal is a two step procedure where they first use an AE to extract common feature representations among tasks and then use the encoder part of it to train a traffic classifier (softmax layer with a softmax activation function) and a traffic predictor (a dense layer with the Rectified Linear Unit (ReLU) activation function) simultaneously. Compared to conventional single-task learning approaches, which do not use AE and tackle classification and prediction tasks separately, the MTL approach always provided the highest performance.

More recently, the authors in [23] proposed a CNN architecture to perform TC without having to decode and/or decrypt any of the transmitted flows. To achieve this, the input to the model are raw physical control channel messages of a mobile network. The input data is obtained by decoding the DCI messages carried within the LTE-PDCCH. Among the information carried by the DCI messages, the authors used the number of allotted resource blocks, the MCS, and the transport block size. The authors claimed that this information should provide sufficient information for learning algorithms to reliably classify the application and the service that the user is running. The results confirmed such claim and the proposed CNN achieved an accuracy above 98%, outperforming other DL models such as RNN and traditional ML approaches such as SVM and RF, Logistic Regressor (LR), k-Nearest Neighbours (K-NN),and Gaussian Processes (GP).

B. TC using L1 classification objects

All the previous works are byte-based approaches, which limits their application on wireless networks (see Section III-A). As a solution, a few spectrum-based traffic classifiers have been proposed in recent years to perform TC on raw spectrum data. Authors in [10] present a DL-based algorithm that can classify traffic patterns of different types of applications directly from the radio spectrum with accuracy $\geq 96\%$ and outperform state-of-the-art methods based on IP-packets with DL. They use images representing the spectrum in time and time-frequency as input data for their CNN-based DL architecture. An extension of this work was presented in [24], where a validation with real-life data showed that a model trained with synthetic data can discriminate between different traffic patterns but with a decreasing performance in terms of accuracy. One advantage of this approach was the automatic extraction of the time-dependent features to perform the TC task. However, in contrast to byte-based methods using statistical data from traffic flows, this approach assumes that the spectrum patterns to be classified belong to a single-user and single-flow, which is not the case in real environments.

An alternative to overcome the limitations of [10] is to use L1 packets and perform the TC directly on them. Based on a RNN architecture, the authors in [8] showed that TC on raw spectrum data could be performed on short time-series (a few hundred samples) with an accuracy $\leq 85\%$. This accuracy can be considered low compared to byte-based TC systems if we also consider that the L1 packets were single-modulated with no coding, non-encrypted, and transmitted with a low data rate. One of the reasons for this performance is the use of RNN architectures, such as LSTMs [25], [26], which suffer from inefficient training and low accuracy with large data sequences [27]–[29].

Given the limitations found in the previous spectrum-based proposals, in this paper we introduce a general framework to achieve TC at any layer in the radio stack. On top of this framework, we propose an end-to-end spectrum-based TC procedure that is also RAT agnostic. At the heart of the procedure, we design, train and evaluate a CNN-based classifier that outperforms an optimized RNN architecture, similar to the one used in [8], by achieving higher accuracy, even on large data sequences, and lower complexity in terms of prediction time. Moreover, the proposed approach removes the need for specialized algorithms for traffic flow separation/aggregation on spectrum data like the one required in [10] as it uses as

classification for TC the IQ-samples associated to one single L1 packet, which can be realized with simpler algorithms. This framework will allow decision-making engines to enhance the view of the traffic that is passing through the gateway nodes, not only traffic from the same network domain (of which we can obtain a byte representation), but also traffic that is generated by any nearby wireless device and captured directly on the shared spectrum. Table II provides a comparison of the contributions of this paper and the analyzed related works that perform TC using DL.

III. A GENERAL FRAMEWORK FOR TC

This section introduces the main limitations of the byte-based approaches for TC in wireless networks using a shared spectrum and presents a general framework to perform TC at any layer of radio network stack using a spectrum representation of a packet. This framework provides the building blocks to design a DL-based traffic classifier for wireless networks. In the rest of the paper, the terms spectrum-based packet representation and L1 packet are used as synonymous, similar to byte- or protocol-based representation and L2 (or above) packet.

A. Limitation of the byte-based frameworks for TC

Deploying wireless networks that can handle the increasing demands on network capacity of new applications and services while guaranteeing their QoS requirements will depend on the radios' capabilities to be aware of their spectrum environment, sharing and re-using it optimally. Therefore, including capabilities to sense and understand the environment state is fundamental to dynamically adapting the radio parameters, so the users' requirements are fulfilled while optimizing the shared spectrum usage [30], [31].

Large deployments of daily use technologies such as WiFi and 4G/5G are based on simple devices at the users' side and a full management stack at the RAT GW node or beyond. One of the management system's key components is the NMS, which provides various information related to the traffic generated by the users accessing the operator's core network. These services have recently been enhanced with ML techniques to perform automatic network traffic analysis with high accuracy [3]–[5]. One of these traffic analysis tasks is TC, which allows inferring the user's application generating the traffic to enforce specific security and QoS policies on it.

To picture the use of TC on wireless networks, let us consider a co-existence scenario in a shared spectrum between two wireless networks, one using Wi-Fi and the other using a private 4G/5G deployment, as shown in Fig. 1. In this scenario, the management system running on the private 4G/5G deployment wants to prioritize the traffic of user 1 over user 2 dynamically, so it can enforce a given QoS while maximizing the use of its network resources. The GW node can perform TC to detect and identify the type of application being executed at any moment by its users. On the other hand, user 3, which belongs to the Wi-Fi network, generates a large traffic volume using a non-priority application. For simplicity, let us assume that a central entity manages both

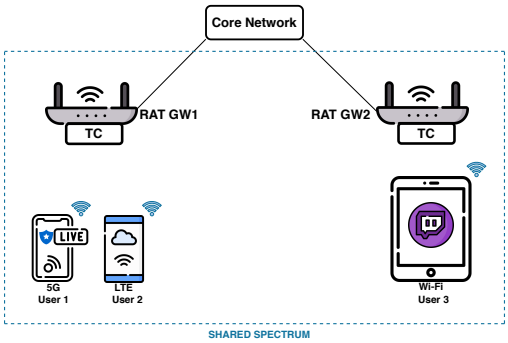


Fig. 1: Use case scenario where the traffic analysis will be incomplete if a byte-based TC system is used.

GWs. However, this can easily be extended to independent management domains where information exchange is allowed to enforce collaboration or/and cooperation among them as we demonstrated in our previous work [7].

Traditionally, byte-based TC systems will be located at (or behind) the RAT GW and will classify the traffic sent by the users' devices in the wireless domain of the RAT GW. In the example scenario, GW 1 can classify traffic from users 1 and 2 and use this information to enforce traffic policies, e.g., GW 1 can determine to reduce the bandwidth assigned to user 2 to guarantee QoS on the protected user 1. However, independently of the traffic policies applied at GW 1 to protect user 1, the performance of user 1 can be negatively impacted by the non-priority traffic generated by user 3. This is expected as the GW 1 can not see the traffic generated by user 3 as it will never pass through it. Even with a byte-based TC system working on all the users' traffic, it would be difficult to infer that the traffic generated by user 3 is being negatively impacting user 1 without knowing the users' location.

B. A TC framework at any layer

One possible solution to the problem described above is that the GW 1 recognizes the radio technology that interferes and adapts its behavior accordingly [7], [32], [33]. However, this approach is not enough to increase spectrum efficiency as

(i) the interfering/interfered technology may not include a mechanism to adapt or coexist, e.g., to change its frequency band to avoid the interference, and
(ii)
(iii) traffic prioritization can be only based on technology and not on application. Therefore, specific traffic policies such as priority vs. non-priority traffic cannot be applied.

For instance, if the GW 2 can recognize that some external device is transmitting a high-priority/protected traffic, it can enforce a policy over its user 3 to reduce the impact on that external device even if it is not under the same management domain as the other wireless device. Moreover, the TC task cannot be performed at L2 or beyond at the central entity, as it requires specialized hardware to demodulate and decode the information of a given technology. Therefore, being able to perform TC to its own users' traffic and any other traffic in

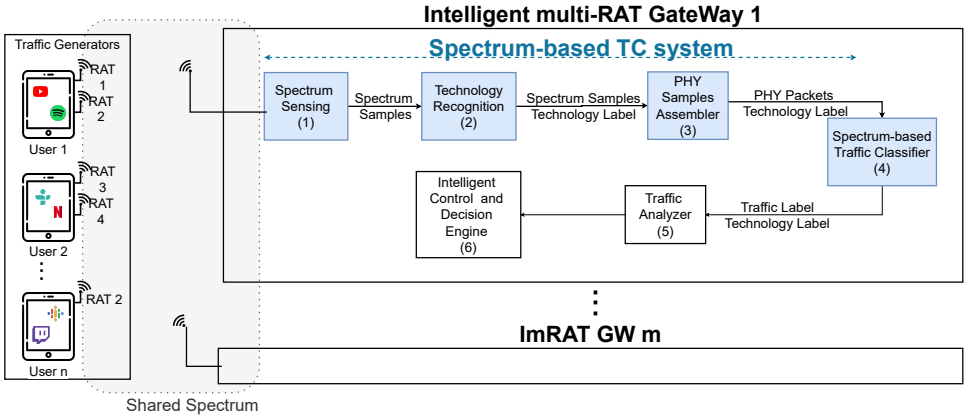


Fig. 2: Functional diagram of a general framework for traffic classification at any layer

the same spectrum is crucial for future deployment of complex and advanced wireless networks in those scenarios.

In general, wired and central-managed wireless networks, using (possibly) multiple technologies, can implement TC at L2 with minor effort as all the users' traffic can be obtained at the access points/gateways of the networks and should be the logical choice for implementing a TC system. However, this task is more complex in decentralized multi-technology wireless environments as the traffic from multiple wireless devices flows through the same shared medium (the spectrum). In the case of multiple wireless technologies, it is required to have specialized hardware to demodulate, decode, and decrypt the messages sensed by the receiver to obtain the L2 packet for TC. In the case of the same technology, each wireless device may belong to different wireless domains, which are usually secured, so packets demodulated and decoded can be encrypted but at the cost of requiring more complex classifiers. As a result, a general framework to classify traffic directly at L1 is key to addressing these wireless network limitations.

To this end, Fig. 2 introduces a general framework that allows the classification of different types of traffic using spectrum representation of the data packet. The proposed framework comprises two main blocks: the traffic generators and the Intelligent multi-RAT (ImRAT) GW. Let us describe more in detail each one of these blocks.

Traffic Generators: In general terms, traffic generators are a combination between a user's wireless terminal and the applications that run on it to generate traffic and are transmitted over the spectrum to the ImRAT GW.

1) *User's Terminal (UT)*: This is a wireless device that runs the applications generating the traffic and transmits it to the ImRAT GW. Although these devices may be very complex and use advanced RATs, e.g., Cognitive Radios (CRs) [30] or Collaborative Intelligent Radios (CIRs) [31], [34], so they can create fully autonomous and distributed networks, in this framework we assume they are simpler devices that are connected to a (multi) RAT GW node that performs the management tasks.
2) *Applications*: This is any software that runs over the UT and generates traffic. One important characteristic of these applications is that the generated traffic follows a pattern at any radio stack level. This requirement is fundamental as any Artificial Intelligence (AI)-based algorithm learning to discriminate among different classes needs to find patterns on the data used to learn. For example, a pattern can be learned from the application's protocol generating users' data traffic.

ImRAT GW: This is a wireless device that acts as a gateway to one or multiple RATs and uses AI-based algorithms to perform wireless management tasks such as the NMS. This device can also be integrated into more advance Radio Access Network (RAN) architectures like OpenRAN [35] or open5G [36]. For deploying spectrum-based TC algorithms, the following internal blocks are required.

1) *Spectrum Sensing*: This sub-module, typically found in CR, obtains samples of the spectrum. The type of data will depend on the input data required by the spectrum-based TC. Some formats of these samples are the In-phase and Quadrature components (IQ) (time domain), Fast Fourier Transform (FFT) (frequency domain), Short-Time Fourier Transform (STFT) (time-frequency domain), Continuous Wavelet Transform (CWT) (time-frequency domain). Note that although traditional radio transceivers do not provide access to the received spectrum data, radio platforms running on Software Defined Radios (SDR) [34], [37] or software tools like Nexmon [38] can already provide access to it.
2) *Technology Recognition (TR)*: This sub-module uses spectrum data to recognize the radio signatures of different RATs and idle (noise) separately [37]. As input, it consumes the continuous stream of spectrum data collected by the spectrum sensing sub-module and outputs if a given technology is present in some part of the spectrum.
3) *Multi-RAT L1 packet assembler*: This sub-module puts together different spectrum samples that belong to the same RAT and creates L1 packets. For this, the labels created by TR are used to find a packet pattern. Note that the spectrum sensing, TR, and this sub-module can be removed if we use the L1 packets already captured by the Wireless Network Interface Card (WNIC) of the multi-RAT before they are demodulated and decoded. However, this implies having one WNIC per technology.

One advantage of the spectrum sensing + TR + Multi-RAT L1 packet assembler is that they are technology agnostic and can be easily extended to support new technologies without requiring adding new WNICs to perform the monitoring task.

4) *Spectral-based TC*: This sub-module uses L1 packets to classify the application that generated it. Compared to the byte-based TC, this approach is more complex and needs to be more robust. The features used to discriminate different traffic classes at different radio stack layers have to be extracted from the raw spectrum. As an example, a user-level application generating traffic will, in general, have a very similar IP packet carrying the application payload. However, the same packet at the L1 will be different depending on properties like the RAT (5G vs. WLAN), its version (WiFi 5 vs. WiFi 6), the MCS (BPSK 1/2 vs. QPSK 1/2), etc.
5) *Traffic Analyzer*: This sub-module takes the output from the TCs and automatically generates an analysis of the traffic flowing in the physical medium. The resulting analysis is used to enhance the decision-making engines' view controlling the radio parameters aiming to optimize a(n) (multi-)objective function.
6) *Intelligent Control and Decision Engine (ICDE)*: This sub-module combines the output of different radio systems, e.g., the output of the NMS and the QoS requirements of a given application/user, and intelligently and dynamically adapts the radio parameters at different layers to improve performance and increase spectrum efficiency. An example of this module is being presented in [31], [39].

Notice that although we did not follow any WLAN or 4G/5G standard to realize the ImRAT, their functional blocks can be positioned inside 5G-RAN architectures like the one proposed by the O-RAN Alliance [35] and coexist/coordinate with WLAN APs in the same network domain as established by the 3GPP release 16 with the introduction of the 3GPP Access Traffic Steering Switching and Splitting (ATSSS) [40]. As an example, the Spectrum Sensing, TR, and the L1 packet assembler blocks can be implemented in the Remote Radio Unit (RRU) for fast signal processing and spectrum-based TC empowered by ML models can be deployed in the RAN Intelligent Controller (RIC) co-allocated Distributed Unit (DU)/Centralized Unit (CU) to reduce the amount of data required to move L1 packets to the classifier. In decentralized environments, where multiple networks may run under their own network management domain, new approaches have to be investigated and explored to define mechanisms to incentive the collaboration among networks and enforce policies that aim the optimization of global objectives shared by the networks in a fully automated fashion. Examples of such collaborative approaches are the Collaborative Intelligent Radio Networks (CIRN), which were developed during the DARPA Gaussian Processes (SC2) competition [41]. Some of our recent work have demonstrated the capabilities of such networks at the architectural level [34], [39] and validated its performance via experimental results [31]. Moreover, theoretical and practical developments of collaborative protocols to support such architecture can be found in [42] and [43], respectively.

This framework allows the deployment of TC systems that work on the spectral representation of transmitted data and realizes a general approach for TC for wireless networks at any layer from L1 to the Application Layer (L7). This framework is general as the first two blocks (technology recognition and Physical-layer (PHY) packet assembler) provide a mechanism to be wireless technology agnostic, while using an L1 packet as classification object allows a classification at any layer as this object contains the whole information carried by the transmitted packet. For end-to-end designing, training, and deployment of the ML/DL algorithms behind the sub-modules like TC and TR, this framework can be enhanced by the ideas presented in [3] (byte-based TC) and [37], respectively.

IV. Spectrum-based TC system based on DL

TC at spectrum level using an L1 view of the packets enhances the traffic statistics provided by an NMS by including information from any traffic flowing in a shared medium such as a wireless link. However, L1 packets carrying the same payload can be completely different due to several factors like:

- Different RATs may use various schemes to transmit (e.g., OFDM vs. DSSS) and modulate/code the data (BPSK vs. QAM).
- Different MCSs produce different packet length carrying the same data.
- Same RATs may have different L1 versions, and therefore each version may have its specification (802.11n vs. 802.11ac).
- The L1 layer may encrypt the data before transmitting.

This heterogeneity makes the TC at the spectrum level an arduous task. In fact, it is almost impossible to use ML algorithms that rely on any feature engineering. Here is where DL plays a fundamental role: this ML technique allows the automatic feature extraction on hyper-dimensional data [44]. These capabilities have also been used in the networking domain to perform classification tasks on high dimensional data like spectrum samples for modulation classification [45] and technology recognition [37], [46], or raw bytes and images representing packets or packet flows at L1 or above for TC [6], [10], [47].

As we have shown in Section II, L1 packets (or packet flows) can be treated as 2D time-series or images representing time-series. Concerning time-series data, there are DL architectures based on RNN, such as LSTMs [25], [26] and Gated Recurrent Units (GRU) [48], [49], that are designed to exploit the structure of sequential data. However, they suffer several drawbacks such as difficulties to be trained and low accuracy with large data sequences [27]–[29].

One way to address the limitations of RNN architectures on large data sequences is to use CNNs for performing the automatic feature extraction while shortening the input sequence length before it is fed into the RNN architecture [19]. However, empirical evaluations have shown that even the recurrent layers are no longer needed for capturing the

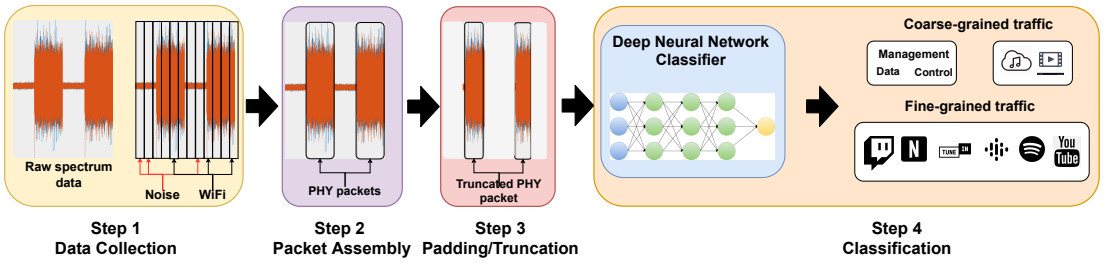


Fig. 3: TC system using spectrum data. 4 steps compose the system: a) data collection, b) L1 packets filtering/assembly, c) zero padding or data truncation of the time series, d) Fine- or coarse-grained classification of traffic

time-series patterns with CNNs [29]. Assuming a generic approach for the selected DL architecture for the TC at the spectrum level, Fig. 3 shows a spectrum-based TC procedure built on top of several of the functionalities presented in the general-purpose framework for TC described in Section III. The procedure comprises four main steps: data collection, L1 packet filtering/assembling, zero-padding/truncation, and classification.

- *Data collection:* In this step, the algorithm continuously captures spectrum samples and pre-processes them before being fed to the DL model. The first process to run in this step is the spectrum sensing to capture the samples in a given format, such as IQ or FFT samples. These samples are then normalized and grouped, for example, by using a fixed-sized moving window. The fixed-size samples are then labeled according to the RAT used to transmit them, the absence of them, i.e., noise, and a mix of them, i.e., interference. This step can be implemented by the spectrum sensing and TR sub-modules of the proposed framework as in [31].
- *L1 packet assembly/filtering:* One crucial aspect of the proposed algorithm is that it assumes that an L1 packet is a self-contained time-series structure where it can find and learn the upper layer protocol's pattern. A combination of IQ samples and labels from the first step provides a mechanism to assembly the L1 packets. More precisely, the IQ samples labeled with a given technology (step 1) are used as a filter to generate different IQ sample flows per technology. Then, the IQ samples labeled as noise are used as a delimiter to assemble them into L1 packets. Cross-correlation with sync words per technology or end-to-end ML approaches for packet detection such as [50]–[52] can be added to increase this step's robustness.
- *Time series Padding/Truncation:* Once an L1 packet has been assembled, or a group/batch of them, we perform the zero-padding, for short sequences, and truncation, for long sequences, to normalize the length of all L1 packets to a given fix value. This step is important as the training and inference speed of DL algorithms can be improved by using sequences of the same length [53] at the cost of increasing the memory footprint. The optimal L1 packet length for padding/truncation is determined while designing and training the TC's DL models. This value will depend on the DL architecture, the technology that generates the L1 packet, and the layer on which the features have to be extracted to perform the TC.
- *Fine- or coarse-grained classification:* In this step, the trained DL model consumes the padded/truncated L1 packets processed in the previous step and classify them. This step can be performed by a unique model or a cascade of simpler models for multi-step classification. Let us use TC on WLAN as an example. An L1 packet can be first classified by a model discriminating between the management, control, and data frames at L2 (coarse-grained classification). Then, the L1 packets classified as data frames can be passed through a second model that classifies them between two types of L7 traffic, e.g., music vs. video (coarse-grained classification). Finally, the samples labeled as music can be further classified according to the mobile application that generates it, e.g., Spotify vs. GPodcast (fine-grained classification).

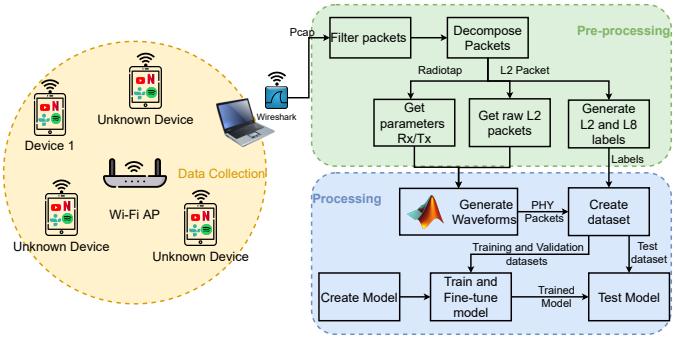


Fig. 4: Hardware deployment and data flow from capturing traffic and dataset creation to model training and validation

The automatic execution of these four steps provides a complete spectrum-based TC system that is also technology agnostic. Of course, several decision choices will depend on the RATs sharing the spectrum and their hardware capabilities to run this algorithm, the channel bandwidth and sampling rate of the sensing module, the traffic classes to be discriminate and at which layer their features can be extracted, and the architecture that is used to build the model(s). However, the proposed framework in Section III and the procedure proposed in this section provide enough flexibility to cover a large number of 5G and beyond use case scenarios using complementary RATs like WLANs and 4G/5G.

TABLE III: Description of the proposed classification tasks to evaluate the spectrum-based traffic recognition approach.

Task ID	Traffic Classification Task	Traffic Classification type	Input representation	Layer on which the task has meaning	Number of Classes	Classes
1	**L2 packet type**	Coarse-grained	IQ samples (Layer 1)	L2	3	Management, Control, Data
2	**L7 Application type**	Coarse-grained	IQ samples (Layer 1)	L7	3	Audio, Video, No application type
3	**L7 Application**	Fine-grained	IQ samples (Layer 1)	L7	7	Netflix, Youtube, Twitch, Spotify, Gpodcast, TuneIn, No application

TABLE IV: Sample distribution per task label and per technology within the task labels (task 1)

Total Samples	Samples per Task Label	Technology		
		802.11b	**802.11g**	**802.11n**
466348	Mgmt: 75156	74662	494	0
	Ctrl: 250967	5340	245627	0
	Data: 72264	5030	337	134858

TABLE V: L1 and L2 packet length stats per label (task 1)

	L1 (Values in IQ pairs)			L2 (Values in bytes)		
	Mgmt	**Ctrl**	**Data**	**Mgmt**	**Ctrl**	**Data**
Mean	25139.37	676.54	4681.98	263.09	21.81	1066.93
Std	3683.24	434.35	3511.11	37.36	7.36	653.55
Min	1600	560	640	30	14	28
Max	37048	4928	131824	397	32	1546

TABLE VI: Sample distribution per task label and per technology within the task labels (task 2 and 3)

Total Samples	Samples per Class Task 3	Samples per Class Task 2	Frames		
			Mgmt	**Ctrl**	**Data**
140665	Spotify: 13822	Audio: 39053	0	0	39053
	Tunein: 10229				
	Gpodcast: 15002				
	Youtube: 16671	Video: 56253	0	0	56253
	Netflix: 18268				
	Twitch: 21314				
	No-App: 45359	No-App-Type: 45359	14805	30554	0

TABLE VII: L1 and L2 packet length stats per label (task 2)

	L1 (Values in IQ pairs)			L2 (Values in bytes)		
	Audio	**Video**	**No-App-Type**	**Audio**	**Video**	**No-App-Type**
Mean	5.77K	10.7K	9.5K	1.2K	1.24K	101.92
Std	5.03K	14.2K	11K	565.77	553.12	123.29
Min	640	640	560	28	28	14
Max	38.9K	138.2K	43.4K	1.5K	1546	579

V. Dataset generation and DL model design

To validate the feasibility of a spectrum-based TC system, we design, implement, and benchmark a DL model based on CNNs. In addition, we also design and implement a baseline model that uses a RNN architecture, similar to the one designed in [8], which is optimized for the provided dataset. The rest of this section will describe how the classification tasks are defined, how we create the dataset to train the models in the specified tasks, and the DL architecture design. This section focuses on the classifier as this is the core of the procedure proposed in Section IV.

It is important to notice that the first two stages of the proposed procedure (data collection and packet assembly) are implemented in an off-line fashion for model training and validation via the dataset creation (see Section V-A). This decision was made since 1) some of our previous works have already shown prototypes that demonstrate is realization [31], [32], 2) the offline data generation provides a more flexible approach to evaluate the feasibility of performing TC directly on L1 packets, and 3) the proposed classifiers (see Section V-C) have as input a single L1 packet, so time-related features that can be extracted by processing spectral data streams are not required while the data storage requirements are minimized in comparison to [10]. The impact of the third stage (time-series padding/truncation) will be evaluated in Section VI.

A. L1 packets Dataset generation

Generating a dataset for spectrum-based classification is a difficult task. However, we follow an approach that uses real L2 packets to generate L1 packets in the form of IQ samples, an approach similar to the one proposed in [10]. Without loss of generality, we selected the 802.11 wireless technology for generating the L1 packets. However, the approach described below can also be used to generate L1 packets from other technologies such as Long-Term Evolution (LTE) with the same emulation platform [4].

Our decision on using a mixed approach (real packet traces + emulation platform to generate the spectrum samples) is motivated on the recent efforts of standardization bodies like the ITU [54], where multi-level ML pipelines are expected to be connected to emulation/simulation sandboxes to generate data for training and performing preliminary model testing [55]. This approach is important to support use cases like the one presented in this paper as generating real spectrum data for training the ML models would require the setup and deployment of infrastructure that is hard to obtain in real life (e.g., isolated environments, management and control on radio transmitter/receiver at different layers, mechanism to change the channel conditions, etc.). These sandboxes will provide the required flexibility to generate synthetic data to train and validate ML models while a complete realization of an integrated wireless and ML architecture with closed loops between the ML deployment platform, the sandbox, and the real network will minimize the inaccuracies of the models inside the sandbox and increase the degree of similarity between the sandbox and the real network.

As shown in Fig. 4, we first perform a data collection step. In this step, we deployed an AP with wired connectivity to the internet. It was placed in a closed space (living room of a home) where it shares the same channel with other APs deployed in neighboring houses. Our AP was configured to

[4]https://www.mathworks.com/products/lte.html

TABLE VIII: L1 and L2 packet length stats per class (task 3)

	L1 (Values in IQ pairs)							L2 (Values in bytes)						
	Spotify	**Tunein**	**Gpodcast**	**Youtube**	**Netflix**	**Twitch**	**No-App**	**Spotify**	**Tunein**	**Gpodcast**	**Youtube**	**Netflix**	**Twitch**	**No-App**
Mean	5.4K	2.9K	8K	7.3K	10.8K	13.2K	9.5K	1.4K	709.14	1.5K	1.1K	1.3K	1.3K	101.92
Std	2.0K	2K	6.9K	7.1K	12K	18.8K	11K	460.12	675.67	263.5	544.84	545.87	550.61	123.29
Min	640	640	960	640	640	640	560	28	28	78	28	28	28	14
Max	38.9K	38.9K	38.9K	65K	138.2K	138.2K	43K	1.5K	1.5K	1.5K	1.5K	1.5K	1.5K	579

use 802.11n standard, with legacy compatibility, on channel 1 (2.4GHz) with 20 Mhz of available bandwidth. Connected to this AP, a mobile device was used to run several L7 applications to generate traffic. Other wireless devices were also connected to the same AP, but they were not managed and might be generating traffic.

This setup provides an easy-to-deploy mechanism to obtain real traffic that is both affected by traffic generated by other wireless devices on the same channel and a large number of variations of the 802.11n protocol stack such as MCS adaptation, L1 diversity (b, g, and/or n due to legacy compatibility of the AP), and L2 packet diversity. Then, a sniffer node (laptop) was used to capture packets over the air without being associated to any WLAN. The captured packets were encrypted as our test network and networks around it were secured (mainly using WPA-2). The collected data were stored in pcap files [5]. Each of these files is named such that we can later on retrieve the name of the program/application generating the traffic. The packets in these files create an intermediate dataset.

The resulting pcaps were then passed to the pre-processing step. In this step, the L2 packets were filtered to remove non-802.11 packets or packets that could not be accessed by the library used to read the pcap files. On the filtered packets, each packet is decomposed into the Radiotap header[6] and the L2 frame. The Radiotap header, which the host machine adds, is used to obtain the physical layer parameters used by radios to transmit/receive the packet. We extracted some information from the L2 packet to generate the labels associated with the L2 type of packet (Management, Control, and Data flags), and then it was converted to raw bytes. All the captured packets are labeled according to the application/protocol that generated them. As our objective is to show the potential of using a TC system that works on L1 packets, we create labels at L2 and L7. We describe the classification tasks that were defined based on the generated labels in the next subsection.

The L1 dataset was then created by combining the raw L2 packets with the information of the PHY associated with the L2 packet. For this purpose, we use the Matlab WLAN (2020b) toolbox[7] to generate standard-compliant waveforms of the L1 packets. To simulate the effects of an environment like a room in a house or a small office over the transmitted signal, the generated waveform was passed through an 802.11n (TGn) multipath fading channel with a delay profile model-B [56] with Gaussian noise.

The resulting L1 packets have a measured Signal-To-Noise-Ratio (SNR) between 20 and 30dB. The modifications applied to noise-free raw IQ samples, such as adding fading channel effects and Gaussian noise, can be seen as data augmentation techniques. With this approach, it is also possible to generate additional L1 packets with other 802.11 PHY but carrying the same L2 Data Frame and use additional channel conditions. This removes the limitations of creating a dataset with such properties on real environments as it will require a highly isolated environment with programmable radios to set the desirable parameters at different radio stack layers and with controllable devices that generate different environment states where the label of generated packet is known.

To the best of our knowledge, this is the first public dataset that contains 802.11 standard-compliant L1 waveforms for testing traffic classification at spectrum level. The waveforms are generated by different 802.11 technologies (b, g, n), which result in different transmission schemes such as DSSS in 802.11b and OFDM in 802.11g/n, different types of L2 frames (management, control and data), and multiple MCS (modulations such as BPSK and CCK for 802.11b and BPSK, QPSK, 16-QAM, and 64-QAM for 802.11g/n with coding rates of 1/2, 3/4, and 5/6 according to the standard and modulation selected). Moreover, the payload carried by these L1 packets (information at L2 and above) were generated using real traces of L7 application running on a mobile device and connected to a secured 802.11 AP with WPA-2. As a result, the provided dataset is more realistic and complex than the one used in [8], which is limited to High-level Data Link Control (HDLC), a simpler L2 protocol whose unencrypted waveforms are modulated only with QPSK at a unique data rate of 1Mbps. Finally, it is worth to mention that the resulting dataset contains a single L1 packet per sample, which is equivalent to the expected output of steps 1 and 2 of the proposed framework, where each packet is a sequence of IQ samples. This approach reduces the storage requirements for the dataset as any IQ sample that is not part of a L1 packet, e.g., noise, is discarded.

B. Traffic Classification tasks

One of the properties to benchmark our approach is the capability to use L1 packets to classify traffic at different layers and different granularity even if the packets are encrypted. For this purpose, the three classification tasks that define the selected labels are described below. Table III shows a summary of the proposed traffic classification tasks based on L1 packets.

1) Task 1 - L2 frame characterization: In this coarse-grained task, the TC algorithm uses L1 packets to determine if the transmitted packet is a Management, Control, or Data L2 frame in 802.11.

[5]https://gitlab.com/wireshark/wireshark/-/wikis/FileFormatReference
[6]https://www.radiotap.org/
[7]https://www.mathworks.com/products/wlan.html

2) Task 2 - Application characterization: In this coarse-grained task, the TC algorithm uses L1 packets to determine the type of application inside the transmitted packet (e.g., audio or video). As only L2 Data frames carry L7 application data, then the algorithm should also discriminate packets that do not carry data.

3) Task 3 - Application identification: In this fine-grained task, the TC algorithm discriminates between the actual applications generating the L7 traffic.

Table IV shows the number of samples and their distribution in terms of frame type and the physical layer technology used to transmit the packets in the generated dataset for task 1. The total number of samples is 466K, where 16% are Management, 54% are Control, and 30% are Data frames. One interesting characteristic of this dataset is that each type of frame was mostly generated with a different 802.11 physical layer. For example, most of the Management frames were transmitted with 802.11b, which is expected as the APs in 2.4GHz work in compatibility mode and use the oldest technology (802.11b) and lowest MCS to transmit their Beacon frames aiming to increase its visibility and resilience. Table V shows that the length distribution is highly associated with the type of frame in terms of packet length (byte and number of IQ samples generated). While most of the Management frames have a mean of 25K IQ samples, Data frames have a mean of 4.6K, and Control only 0.6K. Compared to the packet length at L2, which is the typical representation used by byte-based TC systems, they differ in several orders of magnitude. In this dataset, the largest L2 packet did not exceed a length of 1.5Kbytes, which is only 1.1% of the largest L1 packet length found in this dataset (131K IQ pairs).

The dataset for task 2 and 3 is composed of 140K samples, where 67,8% of the packets are L2 Data type, while the rest are Management (10.5%) and Control (21.7%) (see Table VI). As a result of the 802.11n encryption, no payload of the Data frames can extract information from higher layers, so traditional approaches like port mapping and DPI will not work. Although tasks 2 and 3 are focused on TC at L7, the proposed TC system uses L1 packets as input. Therefore the class formed by Management and Control packets provides a way to filter L1 packets that do not carry L7 information. Analyzing Table VI, we can see that 27.8% of L1 packets were labeled as Audio while and 40% were labeled as Video during the dataset generation in terms of coarse-grained labels. Similarly, in terms of fine-grained labels, 10% of the L1 packets in this dataset are generated by the Spotify application, 7.2% by TuneIn, 10.6% by Gpodcast, 11.9% by YouTube, 13% by Netflix, and 15.1% by Twitch.

Focusing on task 2, Table VII shows that, on average, L1 packets carrying Audio data are smaller than those carrying Video data. However, this is not the case in L2 representation, where both kinds of packets shared similar statistical properties. Similarly, Table VIII shows that L1 representation has more variation on the packet length distribution than L2 ones. The variations on L1 packet lengths are mainly due to the changes on MCS. For example, 75% of the Data frames in the TuneIn application were using MCS 7, while this number dropped to 20% in Twitch L1 packets. In fact, 70% of the Twitch L1 packets are using MCS values between 3 and 7. This provides an interesting set of dynamic parameters that make this representation of the data challenging to extract features.

Once the dataset is created, the last steps of the processing are executed: model creation, training with validation, and testing, which will be described in the next sub-section.

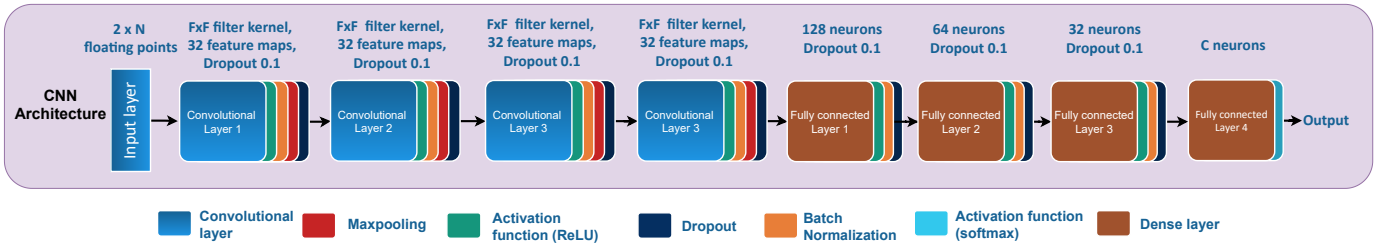
C. DL models design and training

As presented in Section II, the literature is quite limited on TC using raw IQ samples [8], [10]. Thus, to realize the proposed framework to perform TC at any radio stack, we designed and implemented a DL model based on CNNs to overcome the limitations of previous works that either use a RNN architecture [8], or require specific procedures to separate different traffic from different users at spectrum level [10]. As proposed in Section III and realized by the approach presented in Section IV, the object classification for TC models are the IQ values associated to single L1 packet. To the best knowledge of the authors, this is the first time that a CNN is used to solve TC at spectrum level. As baseline, we implemented, fine-tuned and optimized the RNN architecture proposed in [8] to the dataset used in this work. Fig. 5 shows the architecture of the two Neural Networks (NNs) designed for validating our approach.

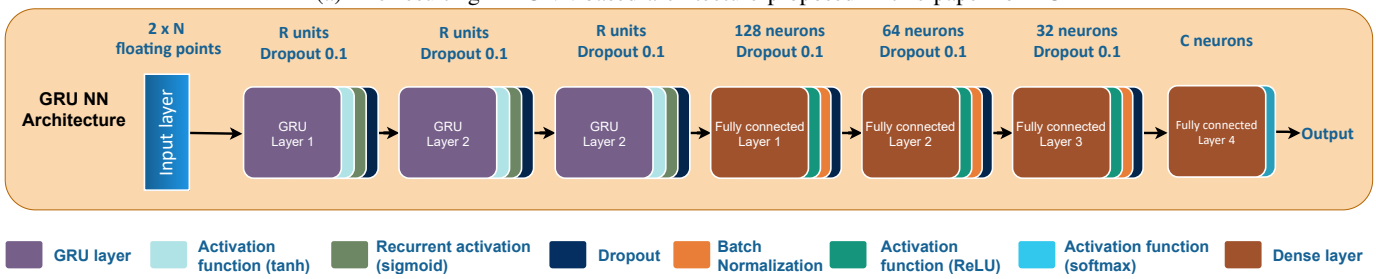
One of the DL models' advantage is their capability to perform automatic feature extraction on raw data to discriminate among multiple classes. In fact, it is that property that allows our approach to perform TC at any layer of the radio stack, e.g., L2 packet type or L7 application type, while using the same input representation. Otherwise, it will be required to use expert knowledge to determine the raw signal features that are useful to discriminate among traffic classes.

The design of the CNN architecture was based on some of our previous experiences solving classification tasks using raw spectrum data such as in [7], [10], [33], [34], [37], [39]. More precisely, we started with a 2D-CNN architecture that worked well in the task of TR with raw IQ samples (see Fig. 4 on [37]), and then we performed a fine-tuning step where the number of Convolutional (Conv) and Dense layers, the number of filters, the filter kernel size of the Conv layers, the maximum (max) pooling windows size, the dropout rate, and the learning rate were varied.

To find the optimal number of Conv and Dense layers, we varied their number between 2 and 4. The lower limit is based on the fact that we need at least 2 layers to learn non-linear functions, and the upper limit is set to 4 as more than that decreased the model's performance on all the experiments. For the number of filters, we tried values in the range between 8 and 128 (in steps of 8). Values above that limit increase the complexity of the CNN to the point that makes it very impractical. To reduce the search space of the optimal configuration, we started by setting the same number of filters on all the layers, and when we found a value that yielded a good performance, we tried to vary this number among the layers. However, keeping the same number of filters on all the layers provided the best results in our case.



(a) The resulting 2D-CNN-based architecture proposed in this paper for TC



(b) The resulting GRU NN-based architecture used as baseline for TC.

Fig. 5: DL architectures designed, implemented and evaluated in Section VI for TC at spectrum level

TABLE IX: Summary of the parameters of the CNN and GRU-NN architectures that were used to perform the different classification task using IQ samples as input data.

Classification Task	Task-dependent parameters				Shared training parameters					
	CNN		GRU NN		optimizer	learning rate	Maximum number of epochs	Batch size	Loss function	Early Stopping monitored function
	Model parameters (N=5000)	Kernel Size (F)	Model parameters (N=5000)	GRU units (R)						
Frame characterization	106K	2	276K	128	adam	0.001	200	64	Categorical cross-entropy	Validation loss
Application characterization	3.72M	32	1.03M	256						
Application identification	3.72M	32	1.03M	256						

With respect to the filter kernel size, we first tried with values in the set $\{2, 3, 5, 7\}$. These values worked well for TR [37] and flow-level traffic recognition using images as input [10]. However, as it will be shown in Section VI, this range did not work well for classifying L7 applications (tasks 2 and 3). Therefore, we increased the range of the filter kernel size and explored values between 8 and 64 (in steps of 8). The results indicated that for solving task 1, the CNN architecture only required an small kernel filter size as it was similar to a TR task. This can be explained as the frame type (task 1) is related to the 802.11 standard used to transmit them as indicated in Table IV. In contrast, tasks 2 and 3 require a larger kernel size to learn helpful information at L7 directly from the spectrum. It is important to recall that increasing the kernel size helps to augment the reception field, which is important in classification tasks with large input sequences [57].

The number of dense layers was also selected in the range between 2 and 4, with decreasing number of neurons from the inner layer towards the output layer with a maximum value of 512 and a minimum value of 16 in the layer before the output. As it can be noticed, we follow the traditional approach of narrowing the network to force it to remove useless information while keeping only the relevant information to reduce computational cost. The resulting CNN architecture is composed of four Conv layers, followed by four dense connected layers. All the layers have ReLU activation functions, except the last one with a soft-max function for classification, and are followed by a dropout layer to improve generalization (reduce overfitting) and a batch normalization layer to accelerate training. Conv layers are also followed by max-pooling layers to down-sample the input. Fig. 5a shows the resulting CNN-based architecture proposed in this paper.

It is important to notice that during the design and fine-tuning phase, we also tried other architectures that have been used to solve classification tasks with time-series data such as CNN+RNN [29] and WaveNet [57], which increases the learning capabilities on long sequences by increasing the reception field without increasing the filter kernel size as it is required in tradition Conv layers. However, they did not provide better performance than the developed model for this paper.

The RNN architecture, designed as baseline, is inspired on [8] with a fine-tuning and optimization steps based on the dataset provided in this paper. Following a similar approach as with the CNN, we varied the number of recurrent layers and the type and number of recurrent units to find the model that performs the best. The number of recurrent layers varied from 1 to 4, achieving the best performance with 3 recurrent layers, result that is aligned with [8]. We also varied the type of recurrent units between GRU and LSTM and we found that GRU had similar or outperformed the LSTM in both execution time and accuracy in all the experiments we run. This result is also aligned with previous findings in other comparative

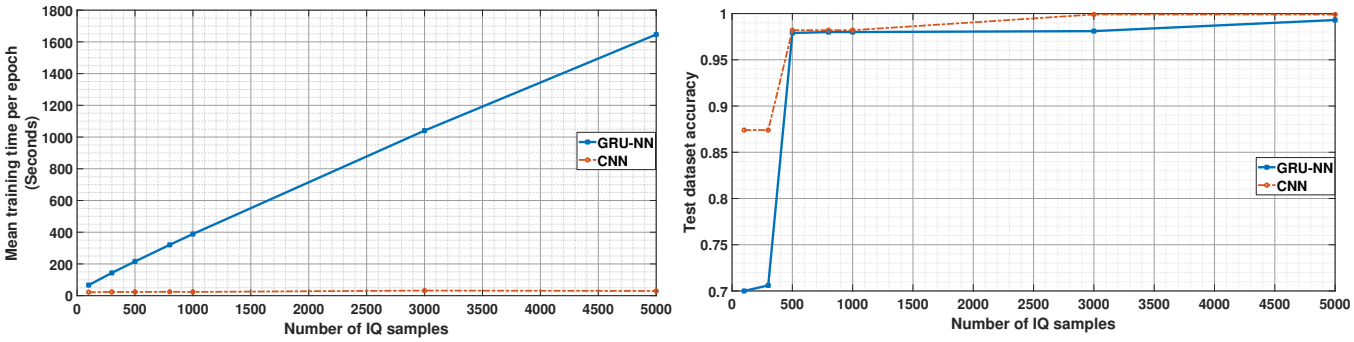


Fig. 6: Training time per epoch (left) and accuracy on the test dataset (right) with respect to the input size N in the task 1

studies such as in [49]. In terms of number of recurrent units, we searched the best value in the range between 64 to 384, in steps of 64 units. The upper limit was set in 384 since larger values generate executions times that are prohibited for real-time classification. Again, the best performance was achieved with values in the set $\{128, 256\}$.

With respect to the dense layers, they have the same configuration as the CNN architecture. This decision was made to provide similar learning capacities in the classification layers between the two models and allow a more fair comparison. We also tried other configurations for the dense layers in the RNN, but the one used in the CNN always provides the same or better accuracy. As indicated above, the three Recurrent layers have GRU units with recurrent activation function sigmoid and activation tanh. This combination of functions allows the use of a fast implementation to improve performance[8]. The Recurrent layers also are followed by dropout layers to improve generalization. Fig. 5b shows the resulting RNN model, which we call GRU-NN, with the same hyper-parameters for the 3 tasks. Similar to the CNN models, the only hyper-parameter that changes among tasks is the number of GRU units (R), where it increases from 128 in task 1 to 256 in tasks 2 and 3.

During the fine-tuning of both models, the hyper-parameters were selected using a multi-round approach of hyper-parameter search over hundreds of executions. The first round was using a reduced version of the task 3 dataset. Then, the resulting CNN and GRU-NN architectures were used as a baseline for another round of hyper-parameter search but using a reduced version of the dataset of tasks 1 and 2. Similarly, it was determined that the best results were obtained using Adam optimizer [58] with a learning rate of 0.001 and a batch size of 64. We use categorical cross-entropy as a loss function and early stopping as a second method for regularization. For each task, a CNN and a RNN model were created. Given the size of the generated dataset and the number of samples per label ($\geq 10K$), we used hold-out cross-validation (i.e., validation with independent test set) to partition the dataset. While the size of the dataset allows maintaining a large number of samples in the training set, we can still guarantee that knowledge about the test set is not leaked into the model so we can also ensure generalization performance. As a result, the dataset was partitioned such that the models were trained with 70% of the samples, while 15% was used for validation and 15% was used for testing. The models were implemented in python, Tensorflow 2.1[9] was used to create, train, and evaluate the resulting models, and the hyper-parameter search was performed using hyperas [10]. The training was accelerated by using Tesla V100 GPUs in our GPULab facility[11].

[8]https://keras.io/api/layers/recurrent_layers/gru/

VI. Results and Discussion

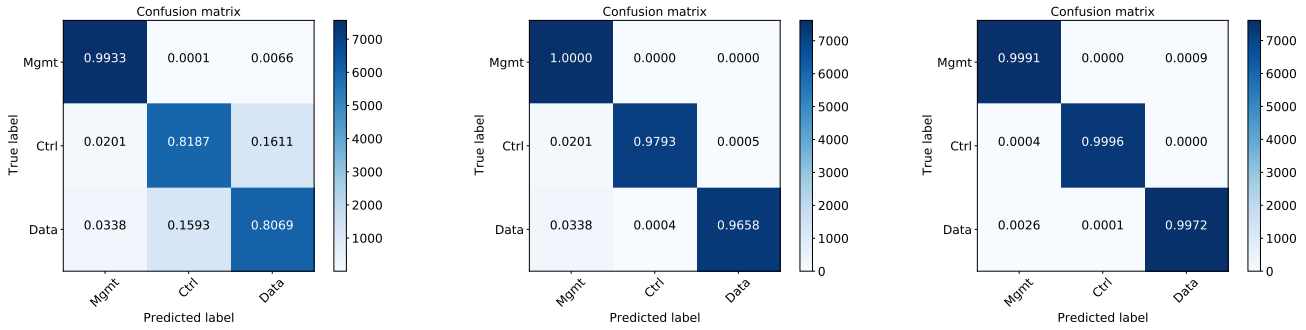
In this section, we present the performance evaluations of the two models proposed in Section V. For the L2 Frame type TC (task 1), we balanced the first dataset by performing under-sampling. As a result, the evaluation dataset contains 75k samples per class, which is driven by the class with a lower number of samples (see Table V). Performing the same operation on the second dataset to generate the samples used for the L7 Application type TC (task 2) and L7 application TC (task 3), the resulting datasets contains 39K samples per class in task 2 and 10.2K samples per class in task 3 (see Tables V and V, respectively).

We run five evaluations on each task, where both models were trained, validated, and tested over different lengths of the input L1 packet in number of IQ sample pairs (100, 300, 500, 800, 1K, 3K, 5K). Notice that when the length of the L1 packet was shorter than the required input length, we apply a zero-padding operation at the end to adjust it (post padding). Otherwise, we truncate it to the required length (post truncation). For each task, we select the results from the best evaluations per model and input size. In the following sub-sections, we will analyze the models' performance in terms of accuracy over the test dataset, training time per epoch, and prediction time per sample. These three metrics are good indicators of how good the model is when classifying unseen data, how costly/hard it is to fine-tune and train the model, and the expected execution time when predictions have to be done over a group of samples.

A. L2 Frame characterization task (Task 1)

In this coarse-grained TC task, the models have to identify if a given L1 packet carries an L2 frame of type Management,

[9]https://www.tensorflow.org/
[10]https://github.com/maxpumperla/hyperas
[11]https://doc.ilabt.imec.be/ilabt/gpulab/index.html



(a) N = 100 (b) N = 500 (c) N = 3000

Fig. 7: Test dataset normalized confusion matrices with different input sizes N (number of IQ samples) in task 1

Control, or Data. This was the most straightforward task among the three proposed in this paper. One indicator is the small filter kernel size, set to 2, or a reduced number of GRU units, only 128, that were required to achieve high accuracy. Fig. 6 (right) shows that even with short sequences, e.g., 500 IQ samples per packet, the models already achieve around 98% accuracy. When the sequence had a length of 3K and 5K IQ samples, the CNN model can outperform the GRU-NN model and achieved an accuracy above 99.9%.

Fig. 7 shows the resulting confusion matrices of different input lengths in the CNN. Interestingly, even with an input length of 100 IQ samples, the models can discriminate with high accuracy the L1 packets carrying Management frames from the other type of frames. At the same time, it still has trouble classifying the Control and Data frames. However, this can be explained as Management frames were mainly transmitted using 802.11b (see Table IV, in which waveform is generated using the DSSS modulation technique, while the Control and Data frames are modulated using OFDM.

Moving to the performance in terms of the training time, we can see in Fig. 6 (left) that the GRU-NN has a training time per epoch that increases linearly to the input length (from 66sec with input length 100 to 1646 sec with input length 5K). Contrary, the CNN training time per epoch increases sub-linear (from 21sec with length 100 to 30sec with length 5K). This result implies that GRU-NNs are hard to fine-tune and make them unfeasible solutions when the problem requires long sequences to improve accuracy. Although this was an expected result, since RNN-based architectures process their input sequentially while CNN can exploit parallel processing, it is interesting to experimentally validate that CNNs outperform RNNs in sequence-to-label classification tasks using radio spectrum data as it has been found in other research fields like speech recognition and voice generation [57].

The final metric to evaluate is how long (on average) takes the model to perform a prediction over a single L1 packet. Table X shows that the CNN can classify an L1 packet containing 3K IQ in 92us, 51 times faster than the GRU-NN. The pcap that generates this dataset includes 466K packets captured in 1193 seconds. It gives us 390 packets per second on average. Using the CNN model, those 390 packets would be classified in less than 35ms. This prediction time is auspicious for its deployment in real-time traffic analyzers.

TABLE X: Prediction time per single L1 packet task 1

Input length (IQ samples)	Model	Average prediction time (ms)
3000	CNN	0.092
	GRU-NN	4.71

Some recent works have shown that similar pre-processing steps on 0.5 seconds of spectrum data, i.e., get spectrum data, framing/packetizing it, and formatting it before sending it to the classifier, can be executed in less than 200ms [31].

B. L7 Application characterization task (Task 2)

This coarse-grained TC task challenges the trained models to classify L1 packets according to the type of application being carried at L7. The three classes to discriminate are Video, Audio, and packets that do not carry L7 application data (No App-Type). When we defined this task, we hypothesized that it should be more challenging than task 1. Our reasoning comes from two facts: first, all L1 packets carrying Data frames were transmitted using 802.11n similar to the task 1 dataset, and 2) the high standard deviation on both Audio and Video L1 packet lengths (see Table VII) would not allow using this feature for discriminating between them. One way to validate our hypothesis is by analyzing the model size and the achieved accuracy, and the impact of the input size on it.

Let us start by analyzing the model size (see Table IX). Note that while task 1 only requires a kernel size of 2, tasks 2 and 3 require a kernel size of 32. The increasing filter kernel size is translated in a larger model, from 106K to 3.7M trainable parameters, and a larger reception field to extract the raw signal features. Here the reception field can be understood as the number of consecutive IQ samples used to represent the learned features. Similarly, the GRU-NN also increases its learning capabilities by setting the number of GRU to 256, so double the number compared to task 1, and moving from 276K to 1.03M trainable parameters.

In terms of accuracy, the CNN model outperforms the GRU-NN model by far, but, in both cases, the achieved accuracies are lower than the ones achieved in task 1, even with a larger input size. More in detail, Fig. 8 (right) shows that while the CNN model was able to achieve 97.8% accuracy on the test dataset with an input length of 3K IQ samples,

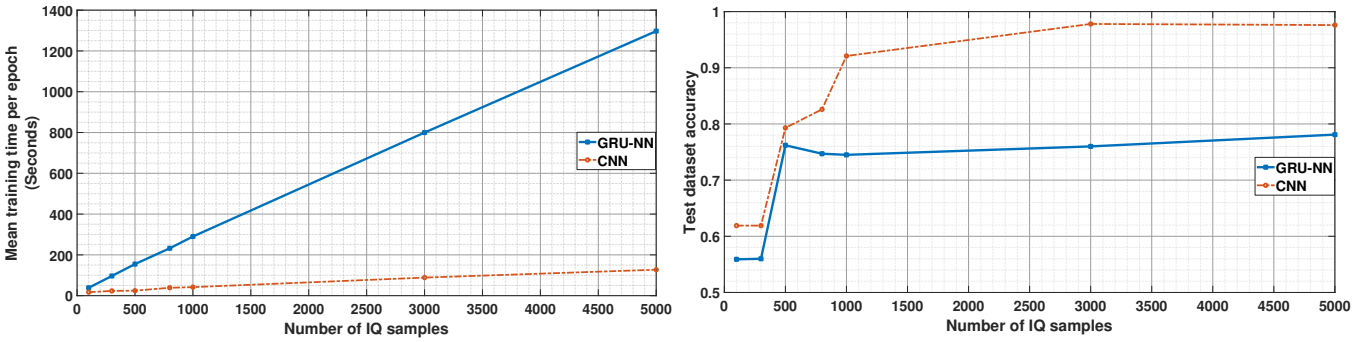
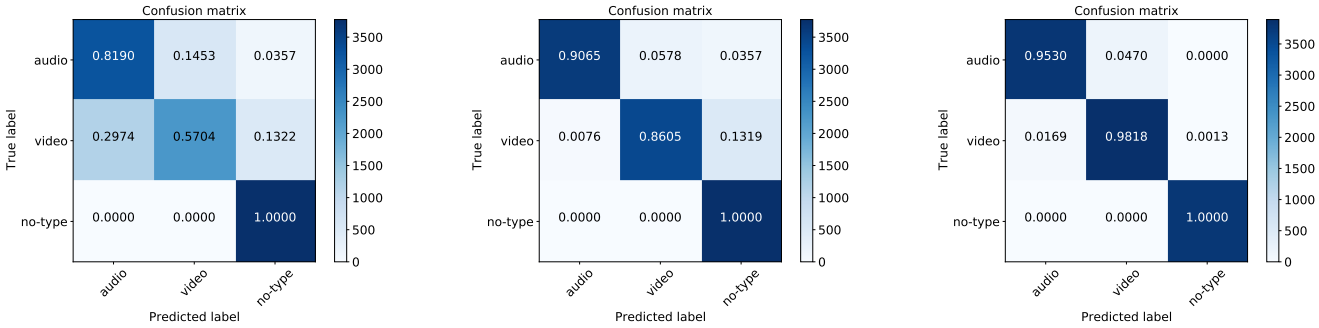


Fig. 8: Training time per epoch (left) and accuracy on the test dataset (right) with respect to the input size N in the task 2



(a) N = 500 (b) N = 1K (c) N = 3K

Fig. 9: Test dataset normalized confusion matrices with different input sizes N (number of IQ samples) in task 2

the GRU could only achieve 78.1% with an input length of 5K IQ samples. In fact, the GRU-NN dropped its accuracy with an input length larger than 500 IQ samples, which was recovered with an input length of 3K, and it was improved at 5K by only 2%. This behavior has also been found in previous works in this area (see Fig. 7 in [8]). It may be possible that the GRU-NN can improve their accuracy with larger sequences, but as we will analyze below, its training cost makes it unfeasible (around 21 minutes per training epoch). Notice that the CNN model achieved 97.6% with an input length of 5K IQ samples, indicating that increasing the input length does not help discriminate among the classes, at least from 3K to 5K. Of course, as the GRU-NN showed it, longer sequences may improve it.

Focused on the CNN, which provides the best performance, Fig. 9 shows that with an input length of 500 IQ samples, the discrimination between L1 packets with Management/Control frames (no application type) is perfect against Data frames (Audio+Video), which is expected based on the results on task 1. However, the CNN model has difficulties separating L1 packets carrying audio and video. With an input length of 1K IQ samples, the CNN model achieved 92.1% accuracy (above 88% if we count only audio and video). With an input length of 3K, this model achieved an accuracy of 97.8% (above 96.7% considering only audio and video classes). This is quite impressive because the input data are L1 packets carrying L2-L7 payloads that were encrypted using the WPA-2 secure method, transmitted with different MCS values, and include simulated channel effects and noise.

Similar to task 1, Fig. 8 (left) shows the linear dependency of the GRU-NN training time per epoch concerning the input length, while this relation is sub-linear with the CNN (from 17sec with 100 IQ samples to 127sec with 5K). However, the slope in the CNN line is higher compared to the results in task 1. This is mainly due to the model's larger size. This result implies that the CNN model is faster to train and fine-tune than GRU-NN, similar to the results in task 1, but it also shows that the CNN model performs much better than the RNN architecture when using longer sequences of data. The better performance of the CNN is also reflected in the prediction time. Table XI shows that the CNN takes 150us to classify an L1 packet containing 3K IQ, 34 times faster than the GRU-NN. If we analyze the pcap of the twitch video app, which is the application with the largest mean packet length (see Table VIII), it contains 21.7K packets, which were captured in 159 seconds. It gives us 136 packets per second on average. Using the CNN model, those 136 packets would be classified in less than 20ms. Again, encouraging results for TC on L1 packets in real-time.

TABLE XI: Prediction time per single L1 packet in task 2

Input length (IQ samples)	Model	Average prediction time (ms)
3000	CNN	0.15
	GRU-NN	5.13

TABLE XII: Comparison of the 3 different approaches for TC on the three evaluated tasks: DL models using raw spectrum and byte representation and Gradient Boosting using the input length as feature

Packet representation	Machine Learning Method	Algorithm	Features	Input Length (Number of Features)	Accuracy		
					L2 Frame characterization	L7 App characterization	L7 App Identification
Spectrum	NN	CNN	Raw IQ samples with zero-padding	3000	**99.86%**	**97.78%**	**90.44%**
		GRU		5000	99.28%	78.10%	54.48%
	Ensemble	Gradient Boosting	Number of IQ samples (L1 packet length)	1	99.06%	80.83%	59.66%
Bytes	NN	CNN	Raw Bytes with zero-padding	1546	**99.99%**	**99.16%**	**94.81%**
		GRU			99.99%	99.12%	94.46%
	Ensemble	Gradient Boosting	Number of bytes (L2 packet length)	1	99.86%	76.35%	54.84%

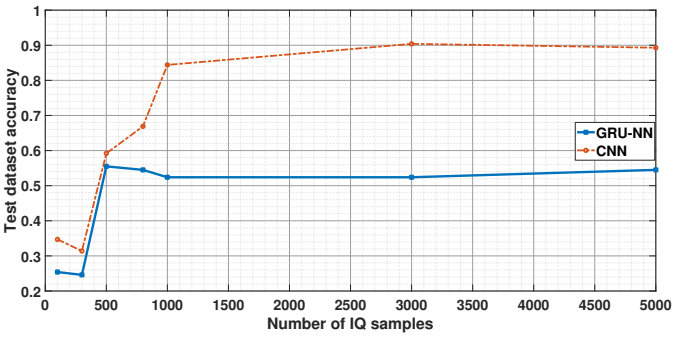


Fig. 10: Accuracy on the test dataset with respect to the input size N in the task 3

C. L7 Application identification TC task (Task 3)

Task 2 is a coarse-grain version of this task, where the models have to classify each L1 packet according to the actual application generating it. In addition to the class that identifies the packets that do not carry L7 application data (No App), there are six other classes: 3 classes of applications that generate video-type traffic (Youtube, Netflix, Twitch) and 3 classes of applications that generate audio-type traffic (Spotify, TuneIn, GPodcast). Section V-B2 and V-B3 provide more detailed information about the dataset used for this task.

This task is useful to determine how much more challenging or easier is the TC if the traffic classes are fine-grained. We hypothesized that this task should be more problematic than its coarse-grained version as the fine-grained applications may have similar statistical properties, and the L7 protocols may also be similar. This is, of course, exploited by the coarse-grained classification, but it can cause troubles to the fine-grained one. This idea comes from analyzing Table VIII, where we can see that while Netflix and Twitch L1 packets have statistical length values very close, they are far from the length stats of audio applications. On the other hand, audio applications have statistical length values similar to each other, which may increase the difficulty of discriminating among them. Notice that this intuition is entirely contrary to the results found in [6], where the traffic characterization task results are better than the application classification.

Fig. 10 indicates that this task is more challenging than the two previous ones. Although the results are very similar to the ones in Fig. 8 (right), there is a drop in accuracy of 7.4% in the CNN (90.4% vs. 97.8% with 3K input length) and 23% in the GRU-NN (54.4% vs. 78.1% with 5K input length) compared to task 2. Focusing on the CNN model results, Fig. 11 shows clearly that the sources of misclassification are located in Netflix vs. Twitch (adding around 19% misclassifications) and Spotify vs. TuneIn ∩ Gpodcast (adding about 17% misclassifications). We omit the results in terms of training time per epoch and prediction time per L1 packet as they are congruent to the results in the previous task as they used the same dataset but with the fine-grained labels.

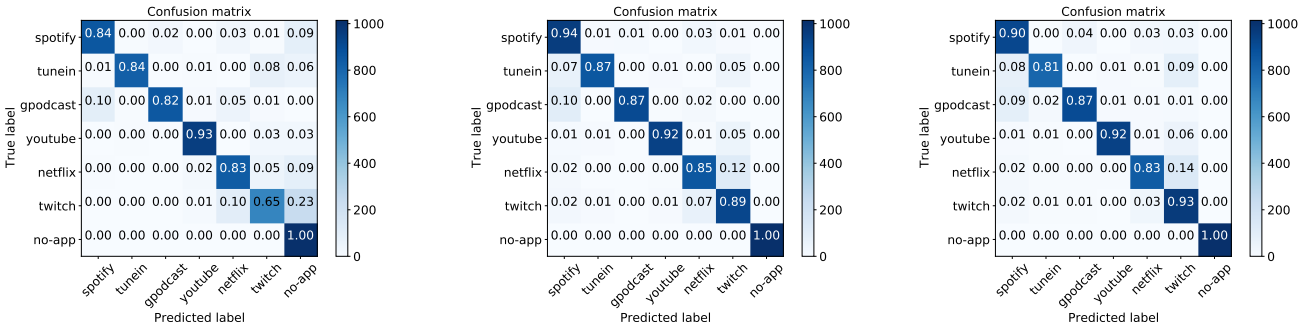
D. Comparison against DL and statistical ML on bytes

To compare the quality of the spectrum-based TC algorithm trained and evaluated in the previous sub-sections, Table XII compares its accuracy against two state-of-the-art DL models that receive a byte representation of the packet at L2, and two ML classifiers, trained with the Gradient Boost (GB) ensemble method. For the statistical ML-based models, we use as unique input feature the packet's input representation length. Note that we can only use this feature for all three tasks as the 802.11n L2 packet's payload was encrypted.

The two DL models for the byte/protocol representation use the same architectures and hyper-parameter configuration as the spectrum-based models presented in Section IV and use a fixed-size input of 1546 bytes, the largest L2 packet captured in the datasets (see Table VII), with post zero-padding/truncation. The GB models were also fine-tuned via hyper-parameter search, and the best results were achieved using a learning rate of 0.1, maximum depth of the individual regression estimators set to 5, 500 number of boosting stages, and 95% of the samples are used for fitting the individual base learners for both models.

As expected, the DL models working on the byte representation of the packet at L2 provide the highest accuracy on all the three evaluated tasks. This can be explained by the well-defined framing at L2 and the shorter length of the input data compared to the dynamic nature of the L1 representation of the input data, where the L1 header has a well-defined structure, but the number of IQ samples representing the payload depends on the channel conditions and the MCS values used while transmitting. This is also reflected in the high and similar accuracy obtained by both the CNN and GRU architectures used in the byte-based approach.

This is the first work that also performs TC on raw bytes from encrypted 802.11n L2 packets to the best knowledge of the authors. Although no pre-processing was done on the raw packets except padding/truncation, the obtained results are aligned with other approaches using DL on raw bytes



(a) N = 1K (b) N = 3K (c) N = 5K

Fig. 11: Test dataset normalized confusion matrices with different input sizes N (number of IQ samples) in task 3

with more pre-processing steps given the protocol structure's visibility on higher layers [6]. However, the spectrum based approach still provides a very competitive accuracy with a similar performance on task 1, a drop of 1.38% in task 2, and 4.37% in task 3, which is outstanding given the added complexity of using spectral data as input, i.e., multi-dimensional, longer sequences, in comparison to an input using raw bytes. In the case of specialized hardware that can demodulate and decode the L1 packets of a given technology, it is clear that the packet-based approach, even on encrypted packets, will be the natural choice to guarantee performance in both accuracy and execution time. However, a realization of the proposed framework removes the need for specialized hardware as L1 packets of any technology can be captured and classified without the need of demodulating and decoding them with a limited negative impact on performance. Of course, the proposed framework opens new challenges on reducing the complexity and improving the performance not only in the classifier but also along the whole chain.

Finally, the GB models show the worst performance, but their results indicate that the input data's length was enough to solve task 1. This result was expected given the high correlation between L2 frame type and the packet length in the dataset (see Table IV). However, this feature is not enough to differentiate the L7 application in tasks 2 and 3. On the contrary, the DL models can automatically extract hidden features from the raw data, probably higher-level protocol structures relevant to solving the other two TC tasks with minimal pre-processing.

VII. Conclusion and Future Work

This paper introduces a general framework to achieve TC at any layer in the radio stack. With this framework, a ImRAT GW would be able perform TC on any packet that is being transmitted over the air in the surroundings of the GW, independent of the technology and the wireless domain which packets belong. We also proposed a novel procedure to perform TC on raw spectrum data on top of this framework. This procedure first combines spectrum sensing with a state-of-the-art approach for recognizing radio technologies to build a PHY representation of the packets. Then, a DL architecture is used to perform TC on the L1 packet. As a result, a unique representation of a single L1 packet is needed to perform the TC at any layer as it already carries all the information describing the different protocols at L2 and above without the added complexity needed to perform demodulation, decoding, and decryption..

To realize the proposed procedure, we focused on building and evaluating two DL architectures to classify the L1 packets. For this, we created and performed a statistical analysis on two different datasets that were used to solve three TC tasks: one coarse-grained at L2 (task 1: frame identification in 802.11b/g/n), one coarse-grained at L7 (task2: Audio/Video/No-App type), and one fine-grained at L7 (task3: 3 Audio apps, 3 Video apps, No-App). The dataset was generated by combining packet traces from real transmissions with a standard-compliant waveform generator for 802.11 radio technologies.

Performance evaluations showed that the DL model based on CNN could achieve the best performance on the three proposed tasks, achieving above 99.9% in task accuracy discriminating among classes in task 1, 97.8% in task 2, and 92% in task 3. These results are very promising if we compared them to byte-based DL models, where spectrum-based achieved similar accuracy on task 1, a drop of 1.38% in task 2, and 4.37% in task 3. Finally, the proposed DL architecture could perform the prediction of a given class in the order of microseconds, prediction times that are compelling for integrating into spectrum-based real-time traffic analyzers.

As future work, there are several challenges to be addressed with the proposed framework and the spectrum-based procedure for TC. First, it is required to evaluate the proposed DL models on L1 packets affected by real channel conditions as a decreasing of performance is expected as demonstrated in [24]. Depending on the results, several mechanisms can be used to minimize the negative impact on accuracy. For example, the mechanism we used to create the dataset, i.e., generating synthetic standards-compliant L1 packets that carry real L2 frames, can perform data augmentation to improve generalization. Second, the evaluations were only carried on the DL models proposed to solve the TC tasks. Although the prediction time was very promising, we only focused on the classification tasks with L1 packets, where the models run on high-end hardware, and we did not benchmark the pre-processing step or compared against state-of-the-art TC

systems on L2. Therefore, it is essential to benchmark the complete procedure in several implementation platforms. This includes SDR-only, SDR+host machine, and RAT on-chip for pre-processing and CPU, low-end GPUs, high-end GPUs for running the models, and their respective comparison against byte-based approaches. Those evaluations will provide an initial base on further improvement on the proposed models such that they can run on constrained-resource devices and find a good trade-off between model size and accuracy while providing competitive results compared to TC systems on L2 packets.

A third open challenge is to reduce the complexity of the proposed approach so that it can be deployed at the network edge and solve the three proposed tasks simultaneously. In this paper, we created three different TC models to solve three different but related tasks. This approach does not scale well at the network edge as it involves a parallel execution of the various models (for training and inference) with the inevitable increment of computational and storage requirements. To overcome this problem, MTL should be explored to reduce the computing/storage requirements, achieve higher performance, and simplify the training procedure. As recently investigated and demonstrated in [21], and combined with distributed learning approaches, such as Gossip Learning (GL) used in [23], MTL might speed up the learning process and increase the system scalability. A related problem is the possible bias and damage on the model performance caused by using fixed-length input as required by CNNs in the presence of significant variations in the input data's length. An alternative to address this problem is to explore RNNs again in the context of MTL (e.g., RNN-based AEs) to reduce the final complexity of the resulting model. However, a careful performance analysis should be carried out to determine if the reduction in computation complexity obtained using MTL compensates the inevitable increase of complexity of using RNNs to support input of variable lengths instead of CNNs.

Finally, transmitting L1 packets over the air will result in receiving packets with a large variety of SNR values compared to the values used in the generated dataset (20 to 30dB). Changes in the SNR values will modify the original signal and, in the case of low SNR values, it will negatively impact the performance of the DL classifier as shown in previous work [37], [45]. For this, we plan to augment the provided dataset with packets generated with a larger set of SNR values (e.g., in the range between -20 to 20 dB) so a SNR sensitivity analysis can be performed. With the resulting dataset, researchers would be able to investigate mechanisms to mitigate the negative impact on the classifier' performance when packets are received with low SNR. For instance, the use of denoising AEs as feature extractors may improve the performance in the presence of high levels of noise. Additionally, the augmented dataset could foster the development and implementation of novel algorithms, closing the performance gap when a classifier is trained with synthetic, but standard compliant dataset, deployed in a real environment.

REFERENCES

[1] M. Roughan, S. Sen, O. Spatscheck *et al.*, "Class-of-service mapping for qos: A statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. ACM, 2004, p. 135–148.

[2] S. Valenti, D. Rossi, A. Dainotti *et al.*, *Reviewing Traffic Classification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 123–147.

[3] G. Aceto, D. Ciuonzo, A. Montieri *et al.*, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306 – 315, 2020.

[4] F. Pacheco, E. Exposito, M. Gineste *et al.*, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019.

[5] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[6] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade *et al.*, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[7] P. Soto, M. Camelo, J. Fontaine *et al.*, "Augmented wi-fi: An ai-based wi-fi management framework for wi-fi/lte coexistence," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–9.

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

[9] S. Riyaz, K. Sankhe, S. Ioannidis *et al.*, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[10] M. Camelo, T. De Schepper, P. Soto *et al.*, "Detection of traffic patterns in the radio spectrum for cognitive wireless network management," in *2020 IEEE International Conference on Communications (ICC)*, Jun 2020.

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

[12] P. Wang, X. Chen, F. Ye *et al.*, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.

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

[14] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[15] Z. Chen, K. He, J. Li *et al.*, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1271–1276.

[16] W. Wang, M. Zhu, J. Wang *et al.*, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 43–48.

[17] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," *arXiv preprint arXiv:1812.09761*, 2018.

[18] X. Wang, S. Chen, and J. Su, "Real network traffic collection and deep learning for mobile app identification," *Wireless Communications and Mobile Computing*, vol. 2020, 2020.

[19] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas *et al.*, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[20] D. Li, Y. Zhu, and W. Lin, "Traffic identification of mobile apps based on variational autoencoder network," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2017, pp. 287–291.

[21] A. Rago, G. Piro, G. Boggia *et al.*, "Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 362–10 374, 2020.

[22] M. Miozzo, Z. Ali, L. Giupponi *et al.*, "Distributed and multi-task learning at the edge for energy efficient radio access networks," *IEEE Access*, vol. 9, pp. 12 491–12 505, 2021.

[23] H. D. Trinh, A. Fernandez Gambin, L. Giupponi *et al.*, "Mobile traffic classification through physical control channel fingerprinting: A deep learning approach," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1946–1961, 2021.

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

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[26] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2.

[27] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[28] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[29] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[30] S. K. Sharma, T. E. Bogale, S. Chatzinotas *et al.*, "Cognitive radio techniques under practical imperfections: A survey," *IEEE communications surveys and tutorials*, 2015.

[31] M. Camelo, R. Mennes, A. Shahid *et al.*, "An ai-based incumbent protection system for collaborative intelligent radio networks," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 16–23, 2020.

[32] V. Maglogiannis, A. Shahid, D. Naudts *et al.*, "Enhancing the coexistence of lte and wi-fi in unlicensed spectrum through convolutional neural networks," *IEEE Access*, vol. 7, pp. 28 464–28 477, 2019.

[33] A. Shahid, J. Fontaine, M. Camelo *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)*. IEEE, 2019, pp. 1–8.

[34] S. Giannoulis, C. Donato, R. Mennes *et al.*, "Dynamic and collaborative spectrum sharing: The scatter approach," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2019, pp. 1–6.

[35] O. R. Alliance, "O-ran: Towards an open and smart ran," *White Paper, October*, 2018.

[36] P. K. Taksande, P. Jha, A. Karandikar *et al.*, "Open5g: A software-defined networking protocol for 5g multi-rat wireless networks," 2020.

[37] M. Camelo, A. Shahid, J. Fontaine *et al.*, "A semi-supervised learning approach towards automatic wireless technology recognition," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Nov 2019, pp. 1–10.

[38] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: Build your own wi-fi testbeds with low-level mac and phy-access using firmware patches on off-the-shelf mobile devices," in *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, ser. WiNTECH '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 59–66. [Online]. Available: https://doi.org/10.1145/3131473.3131476

[39] R. Mennes, J. Struye, C. Donato *et al.*, "Collaborative flow control in the darpa spectrum collaboration challenge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2024–2038, 2020.

[40] M. Boucadair, O. Bonaventure, M. Piraux *et al.*, "3GPP Access Traffic Steering Switching and Splitting (ATSSS) - Overview for IETF Participants," Internet Engineering Task Force, Internet-Draft draft-bonaventure-quic-atsss-overview-00, May 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-bonaventure-quic-atsss-overview-00

[41] DARPA, "Darpa spectrum collaboration challenge," (Visited on 10-August-2021). [Online]. Available: https://www.darpa.mil/news-events/spectrum-collaboration-challenge-sc2

[42] A. Sinha and M. P. Wellman, "Incentivizing collaboration in a competition," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019, p. 556–564.

[43] DARPA, "Phase 3 sc2 cil project," (Visited on 10-August-2021). [Online]. Available: https://github.com/Furtad0/CIL

[44] F. Shaheen, B. Verma, and M. Asafuddoula, "Impact of automatic feature extraction in deep learning architecture," in *2016 International conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–8.

[45] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, Eds. Springer International Publishing, 2016, pp. 213–226.

[46] M. Kulin, T. Kazaz, I. Moerman *et al.*, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, 2018.

[47] Wei Wang, Ming Zhu, Xuewen Zeng *et al.*, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717.

[48] K. Cho, B. Van Merriënboer, C. Gulcehre *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[49] J. Chung, C. Gulcehre, K. Cho *et al.*, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[50] M. Kim, Z. Zhang, D. Kim *et al.*, "Deep-learning-based frame format detection for ieee 802.11 wireless local area networks," *Electronics*, vol. 9, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/7/1170

[51] V. Ninkovic, D. Vukobratovic, A. Valka *et al.*, "Preamble-based packet detection in wi-fi: A deep learning approach," 2020.

[52] D. Magrin, C. Pielli, C. Stefanovic *et al.*, "Enabling lte rach collision multiplicity detection via machine learning," in *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, 2019, pp. 1–8.

[53] M. Dwarampudi and N. Reddy, "Effects of padding on lstms and cnns," *arXiv preprint arXiv:1903.07288*, 2019.

[54] ITU-T, "Architectural framework for machine learning in future networks including imt-2020," ITU-T, Recommendation ITU-T Y.3172, Jun, 2019. [Online]. Available: http://handle.itu.int/11.1002/1000/13894

[55] F. Wilhelmi, S. Barrachina-Munoz, B. Bellalta *et al.*, "A flexible machine-learning-aware architecture for future wlans," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 25–31, 2020.

[56] V. Erceg, L. Schumacher, P. Kyritsi *et al.*, "Tgn channel models," IEEE, Tech Report Version 4, May 2004.

[57] A. v. d. Oord, S. Dieleman, H. Zen *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.