DEEP LEARNING FOR WIRELESS COMMUNICATIONS

MIQUEL SIRERA PERELLÓ

Thesis supervisor: KAUSHIK CHOWDHURY (Northeastern University)

Tutor: ALBERTO CABELLOS APARICIO (Department of Computer Architecture)

Degree: Bachelor's Degree in Data Science and Engineering

Thesis report

Facultat d'Informàtica de Barcelona (FIB)
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB)
Facultat de Matemàtiques i Estadística (FME)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

I am deeply grateful to my two supervisors, Kaushik Chowdhury and Albert Cabellos-Aparicio, for their invaluable insights, support, and guidance throughout this research. I would also like to express my gratitude to Stratis Ioannidis for sharing his expertise in machine learning, which has greatly contributed to the development of this work.

I extend my appreciation to all my lab coworkers for their collaboration, engaging discussions, and assistance during this project. I am particularly thankful to Mauro Belgiovine and Joshua Groen, with whom I have closely collaborated. Their guidance and willingness to help, even amidst competing priorities, have been truly valuable. They serve as exceptional role models for researchers.

I am immensely thankful to the friends and individuals I have had the privilege to meet during my time in Boston. They have made of my stay in this city an unforgettable experience. I would like to express my special gratitude to my two roommates and O.G.P. for creating a sense of belonging and making me feel at home in a foreign country.

I would also like to thank and acknowledge the friends and teachers who have been a part of my educational journey during my bachelor's degree. Their guidance, encouragement, and knowledge have played a paramount role in my growth both as a person and as a professional.

Last but certainly not least, I am deeply thankful to my family for their immense support, love, and belief in my abilities. Their constant encouragement and understanding have always given me the strength to keep working towards my future.

I am truly grateful to all the individuals mentioned above and the countless others who have contributed to my growth and success.

Abstract

CATALÀ:

Aquest treball avalua el rendiment de diversos models d'aprenentatge profund per a la classificació d'ones en diversos entorns de canal i condicions de soroll. Els nostres resultats demostren la superioritat de les arquitectures basades en *Transformer* sobre enfocaments establerts com les CNNs en una gran varietat de configuracions. Aquests models han sigut entrenats amb dades sintètiques i augmentats en diversos escenaris per aconseguir una generalització robusta. Analitzem àmpliament l'impacte de condicions de canal desafiants i del soroll en les transmissions sense fil i en la predicció. La comparació entre arquitectures mostra els seus avantatges, desavantatges i adequació en termes de precisió, mida del model, temps d'inferència i robustesa. Aquesta investigació obre vies per a més exploració i aplicacions pràctiques, mostrant la integració sense fissures de l'aprenentatge automàtic per a serveis en temps real com és la identificació de protocols Wi-Fi en comunicacions sense fil.

ESPAÑOL:

Este trabajo evalúa el rendimiento de diferentes modelos de aprendizaje profundo para la clasificación de ondas en varios entornos de canal y condiciones de ruido. Nuestros resultados demuestran la superioridad de las arquitecturas basadas en *Transformer* sobre enfoques ya establecidos como las CNNs en una amplia variedad de entornos. Estos modelos fueron entrenados con datos sintéticos y aumentados con varios escenarios para lograr una generalización robusta. Analizamos exhaustivamente el impacto de las desafiantes condiciones de canal y ruido en las transmisiones inalámbricas y en la predicción. La comparación entre arquitecturas revela sus ventajas, desventajas e idoneidad en términos de precisión, tamaño del modelo, tiempo de inferencia y robustez. Esta investigación abre vías para más exploración y aplicaciones prácticas, mostrando la integración sin costura del aprendizaje automático para servicios en tiempo real como la identificación de protocolos Wi-Fi en las comunicaciones inalámbricas.

ENGLISH:

This work evaluates the performance of deep learning models for waveform classification in various channel environments and noisy conditions. Our results demonstrate the superiority of Transformer-based architectures over established approaches like CNNs in a wide variety of settings. These models were trained with synthetic data and augmented with diverse scenarios to achieve robust generalization. We extensively analyze the impact of challenging channel and noise conditions on wireless transmissions and correct inference. The comparison of architectures reveals their advantages, trade-offs, and suitability in terms of accuracy, model size, speed, and robustness. This research opens avenues for further exploration and practical applications, showcasing the seamless integration of machine learning for real-time services like Wi-Fi protocol identification in wireless communications.

Keywords

Wireless communications, Machine Learning, Deep Learning, Transformers, Convolutional Neural Network (CNN), Data augmentation, Protocol identification, Wireless channel, Wi-Fi waveform.

Contents

1. Introduction and state of the art

This bachelor thesis presents the work developed by the Universitat Politècnica de Catalunya Data Science and Engineering student Miquel Sirera Perelló under the supervision of Prof. Kaushik Chowdhury (Northeastern University) and Prof. Albert Cabellos-Aparicio (Universitat Politècnica de Catalunya). The task we will be tackling consists of applying deep learning methods in order to perform inference over waveforms, more specifically we will be classifying the protocol with which a signal has been generated through different types of Neural Network (NN) architectures acting as the classifiers. The data used for classification comprises Wi-Fi collected waveforms under four different types of protocols. We present a thorough comparison between the architectures presented as well as an analysis of its benefits and drawbacks in terms of classification accuracy, inference time, size of the model and input size required to predict. Specifically, we will evaluate various convolutional neural network-based (CNN) and Transformer-based models.

We also conduct an evaluation of how various environmental conditions affect the transmitted data and examine their impact on the process of protocol identification. To assess this, we gradually expose the models to a wider range of conditions, thereby enhancing their ability to generalize in our testing environment. This investigation enables us to gain insights into the robustness and adaptability of the models in different scenarios, providing valuable information for real-world applications.

1.1 Wi-Fi protocols description

As stated, we will be classifying Wi-Fi waveforms. Wi-Fi, short for "Wireless Fidelity", is a technology that allows the connection between devices and the internet or the communication between each other in a wireless manner over a local area network (LAN). It is a wireless networking standard based on the IEEE 802.11 family of protocols. Wi-Fi has made it possible for all kinds of devices, such as smartphones, laptops, tablets, smart home devices, and others, to connect to the internet without the need for physical wired connections. Wi-Fi typically operates at the 2.4 GHz or 5 GHz frequency bands. This technology, which works through radio waves, allows for communication and high-speed data transmission between devices within the area covered by an access point or Wi-Fi network router.

Several standards or Wi-Fi protocols have been developed over the years. These protocols define the specifications for wireless communication and ensure compatibility between different devices that support Wi-Fi. In this work, we will focus on classifying four commonly used Wi-Fi protocols:

(1) `802.11b`: This was the first widely adopted Wi-Fi protocol, introduced in 1999. It operates in the 2.4 GHz frequency band and can offer a maximum data rate of 11 Mbps.

(2) `802.11g`: This protocol, introduced in 2003, also operates in the 2.4 GHz frequency band and supports data transfer rates up to 54 Mbps. It uses the same bandwidth as 802.11b and is fully backward compatible with it.

(3) `802.11n` (also known as Wi-Fi 4): Introduced in 2009, 802.11n was the first Wi-Fi standard to support multiple-input multiple-output (MIMO), which greatly increased data throughput. It can operate in both the 2.4 GHz and 5 GHz frequency bands. 802.11n offers improved speeds and range compared to previous protocols, with data transfer rates up to 600 Mbps. Additionally, 802.11n devices can be backward compatible with older protocols.

(4) `802.11ax` (also known as Wi-Fi 6): Adopted in 2019, this backward-compatible protocol operates in both the 2.4 GHz and 5 GHz frequency bands. It offers increased speeds, capacity, and efficiency compared to previous protocols. Wi-Fi 6 supports data transfer rates up to several Gbps and is specifically designed to enhance throughput-per-area in high-density environments with numerous connected devices.

These protocols possess distinct capabilities and demonstrate the technological advancements over the years. The Wi-Fi protocol supported by a device determines its maximum data transfer rate and its compatibility with other Wi-Fi networks and devices.

1.2 Protocol identification

The problem faced involves the classification of the waveforms that have been described, in different channel conditions and noise levels that will be described in further detail later on.

Protocol identification is usually performed looking at a specific part of the transmitted signal named the preamble. This is placed at the beginning of the transmission and allows the receiver to know how the data is transmitted so that the communication can be effective.

In the case of Wi-Fi signals, each specific Wi-Fi protocol incorporates distinct patterns and synchronization signals in the preamble, which play a crucial role in symbol timing recovery and frame synchronization. Through analysis of this preamble section in the transmitted signal, the receiver radio is able to detect and interpret these patterns, thereby identifying the specific standard used for data transmission. The unique structure and synchronization sequence inherent to each Wi-Fi standard enable the receiver to differentiate between protocols such as the ones we are utilizing (802.11ax, 802.11b, 802.11n, and 802.11g).

One of the main points of the presented solution, is that we will not be looking for the preamble but just giving the raw received signal to our classifier in order to identify the standard being used at a specific moment. This means that the network will have to learn the differences in the data to perform identification and not how to decode any specific part. The identification will happen over samples that contain the preamble, but also over parts of the data that in principle do not contain any of that information.

The solution will consist on two neural network architectures, namely a convolutional neural network (CNN) and a Transformer-based network acting as the two model types that will be trained to identify which of the protocols that has been described is the one that is being transmitted at a certain moment. It is very important to notice that the classifier should be able to generalize without loosing performance. The performance of our classifiers will be evaluated through classification accuracy.

1.3 Machine learning for wireless communications

Wireless communication refers to the transfer of information over a distance without the use of electrical conductors or wires. Instead, it relies on electromagnetic waves, such as radio waves to transmit information from one place to another. It is used in a wide range of applications, including cell phones, radio and television broadcasting, satellite communication, and wireless networking.

Machine learning itself has been a point of interest in many different fields such as natural language processing (NLP) with the current raise of large language models or image and video processing, for tasks like classification, segmentation, denoising, etc. In recent years, the field of telecommunications, including wireless communications, has taken interest in using machine learning techniques to solve some of the challenges that they face. It has been applied in a variety of contexts such as for different tasks managing the emerging concept of Open Radio Access Network (O-RAN), in security enhancing with ideas like radio fingerprinting, beamforming optimization or inference at the edge. This last topic corresponds to the work that will be developed in this thesis.

Specifically in this work we will be focusing on Wi-Fi transmitted waveforms with the objective to perform protocol identification after receiving any kind of transmission. This responds to the interest of being able to introduce ML modules that can perform real- or near real-time inference over some arriving signal.

1.3.1 Genesys lab at Northeastern University

This thesis has been developed at the `Genesys - Next GEneration NEtworks and SYStems Lab`, a laboratory group inside the Institute for the Wireless Internet of Things at Northeastern University. The director of this laboratory is Prof. Kaushik Chowdhury. They have devoted big efforts in pushing the state of the art inside wireless communications, with focus in recent years on applying Machine and Deep Learning solutions to wireless communications problems.

It is widely believed that Artificial Intelligence (AI) will play a key role in the next generation of wireless devices. These technologies that come after 5G, will include AI decision blocks in very important research topics such as resource allocation, security enhancing, data rate improvement and signal classification. This work will only focus on the former one.

AI solutions have emerged as groundbreaking tools in addressing conventional communication challenges, ushering in a revolutionary era of communication referred to as next-gen communications. This paradigm shift has prompted the Genesys lab to diligently allocate substantial resources towards the exploration of myriad possibilities. With the field of Machine Learning (ML) is continuously evolving and improving, the lab is committed to harnessing the full potential of AI and ML techniques to unlock innovative approaches and a comprehensive toolbox that redefine the landscape of modern communications.

1.4 IQ samples transmission and collection

We will be working with samples at the physical layer to identify the protocol with which a waveform is being transmitted, namely, IQ samples. The in-phase (I) and quadrature-phase (Q) samples are used to represent a modulated radio frequency (RF) waveform. They are composed of a real signal (I) and a complex one (Q) that in conjunction will serve to modulate the desired signal.

IQ samples are a representation of the data in the time domain, that is normally changed into the frequency one in order to decode the information that has been transmitted through the frequency or amplitude modification of a base signal.

In a real life setting, this desired signal will be transmitted by a transmitter on one side and and received by a receiver on the other one. When these signal or waveform is going through the air, acting as the channel, it can suffer modifications that will affect the quality at the receiver side (making the classification task harder). Environmental effects such as humidity, being in a closed or open space or the distance between the two antennas transmitting and collecting data can have a decisive impact on what is received. This poses an important problem that will be further explored.

The data that will be used for this project will not be obtained from a real setting, it will be collected through `MATLAB` and the just described channel effects will be added through already existing and specific software. Preexisting synthetic wireless frameworks allow us to train and test solutions without the expensive task of collecting real data, being understood by real data the one collected by the transmission of data from a transmitter to receiver through any kind of channel.

Despite the fact that the model will be trained and tested with synthetic data, the aim of this work is that the found ML solutions are able to generalize in a manner that after their deployment on a real receiver, the models are still able to display good classification accuracy.

1.5 Previous work

There was some background work previous to this thesis that consisted in reproducing an state-of-the-art paper in ML for radio frequency fingerprinting (RF fingerprinting). This was done within the subject Introduction to Research in Data Science and Engineering (I2RCED) during the fall semester of 2022. The specific paper reproduced was *ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks* [11], in which they sought to identify radio signatures through IQ samples at the physical layer.

The process of radio frequency fingerprinting (Fig. 1) can be defined as the process to recognize the device which is transmitting a radio waveform by only looking at the properties of the transmission itself. RF fingerprinting is possible due to the fact that a radio device will introduce very small modifications to a signal being transmitted due to little variations on the hardware (HW). This means that even two radios corresponding to the same manufacturer and model and transmitting the same signal, can be identified as different thanks to these so called HW impairments. By only focusing on the radio signatures that will not vary with mobility or environmental changes we should be able to identify devices even in changing conditions. The ML classifier will learn specific patterns that despite being affected by the channel of the transmission, should be good enough to predict the correct class by the classifier.

It needs to be pointed out that this was not always true as some channel variations can indeed hide the signature of the devices, as well as in our case the protocol being transmitted. Channel conditions are the most important problem that needs to be faced by this kind of ML modules with the intention of performing inference on real deployments.

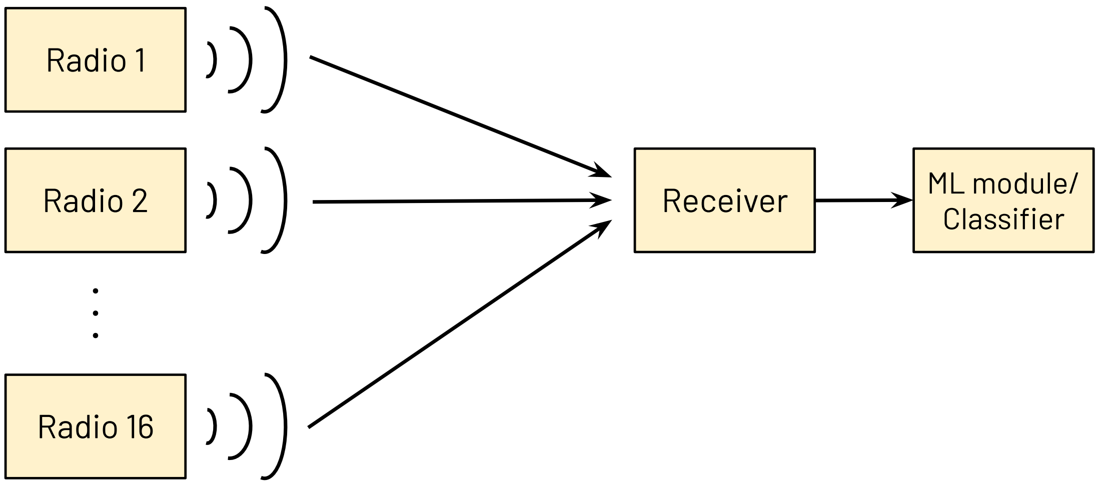


Figure 1: Radio Frequency fingerprinting schema.

The results obtained were highly similar to those obtained in the original paper. We were able to validate and support the conclusions drown from the ORACLE paper. We were also able to observe how the channel variations affect more to the signal than the imperfections that the radio hardware might introduce. Being the channel in this problem and the one faced in this thesis an important challenge to deal with.

It is relevant to the work presented because it is an example of a machine learning classifier, in this case a CNN performing inference over the same kind of data. While we were identifying the device through IQ samples in I2RCED, now we will be identifying the protocol of the waveform being received. It also helps demonstrate the interest in research and industry for the problem of adding ML modules in the telecommunications loop.

The other reason the previous work is highly relevant is because the architecture used for the convolutional network is very similar to the one that will be used in this thesis, being the only difference the input size or the amount of IQ samples we will be using in order to predict which protocol is being used.

In the current project, the CNN architecture is considered as a baseline, since it was already implemented and used in other similar papers in the lab group such as [11], the one we already explained on RF fingerprinting or [12], in which this type of neural network is used to decode a modified waveform to avoid having to transmit a part of the preamble in a signal.

1.6 State of the art

In this subsection, we provide an overview of different deep learning methods use in several RF problems. The problems we provide examples for are modulation classification, RF fingerprinting, beamforming or semantic communications. These are classical problems inside the field of telecommunications and solutions have been explored using machine learning methods.

An intriguing observation lies in the widespread utilization of Convolutional Neural Networks (CNNs) [11, 7, 9] as a prominent architecture in machine learning applied to wireless communications problems. Within the domain, CNNs have emerged as the primary and dependable choice for numerous challenges. The inherent capability of this architecture to preserve and extract spatial relationships through the usage of kernels proves immensely advantageous across a multitude of presented problems. By leveraging these kernels, CNNs aptly capture and analyze the intricate spatial features embedded within wireless communication datasets, facilitating enhanced performance and accuracy. The prevalence and success of CNNs in addressing wireless communications challenges underscore their significance and efficacy as a very important tool in this domain.

Despite the widespread use of this neural network architecture, there have been observations regarding opportunities for improvement. Consequently, alternative architectures have been explored due to their superior adaptability to specific problems or particular properties that render them more suitable in certain contexts. Auto-Encoders, as in [13], have demonstrated exceptional performance in learning compressed representations and have greatly contributed to numerous denoising applications. Another captivating tool are generative adversarial networks (GANs), as highlighted in [10]. Adversarial training with GANs presents intriguing prospects for augmenting wireless datasets, which can be prohibitively expensive to collect in real-world settings. This architecture serves to bridge the divide between synthetic generated data and real-world data, enabling the generation of synthetic samples that closely resemble real-world data.

The emergence of Transformers in recent years has presented an intriguing approach to addressing several challenges with a fresh perspective. Their application in various contexts, as demonstrated in [4, 16], has showcased the remarkable capabilities of this architecture even in highly adverse conditions characterized by the presence of high noise or challenging channel environments. One of the defining features of Transformers is the attention mechanism, which is explained in further detail later in this thesis (3.3.1). This mechanism empowers the model to obtain enhanced representations of the input data, thereby

Paper	RF Application	Architecture	Contributions
[11]	RF fingerprinting	Convolutional Neural Network (CNN)	The use of a convolutional neural network to identify radios in a large pool of bit-similar radio devices, and also the introduction of impairments to enhance classification.
[15]	RF fingerprinting	Recurrent Neural Networks (RNN)	The use of long short-term memory based RNNs to perform transmitter classification with signal to noise ratio down to -12 dBs.
[10]	RF fingerprinting	Generative Adversarial Networks (GANs)	Using this kind of architecture they identify rogue RF transmitters and classify trusted ones.
[7]	Beamforming	Convolutional Neural Network (CNN)	Using a CNN at the transmitter side in order to optimize user equipment performance, handling channel evolution as well as the beamforming phase.
[2]	Beamforming	Multi-layer perceptron (MLP)	Introduces a machine learning module to learn the optimal beamforming strategy by training on received sequences which represent the achievable rates.
[13]	Beamforming	Auto-Encoder (AE)	An hybrid beamforming approach based on an Auto-Encoder neural network achieving better performance in terms of bit error rate than other competitors.
[9]	Modulation classification	Convolutional Neural Network (CNN)	The use of a CNN to perform automatic modulation classification avoiding the requirement that other methods presented of manually extracted features.
[4]	Modulation classification	Transformers	The Transformer architecture is used for automatic modulation classification, exploiting the attention mechanism to obtain better results even in higher noise conditions.
[16]	Semantic transmission	Transformers	Redesign of the Vision Transformer [5] as a backbone in order to perform semantic image transmission to obtain better end-to-end transmission through a wireless channel.

Table 1: Overview of the state-of-the-art ML methods for wireless applications.

boosting performance and classification accuracy in diverse problem domains, including modulation and protocol classifications, mirroring the focus of our current project.

Finally, it should be pointed out, that another interesting telecommunications problem that has been tried to solve with ML is wireless localization. In [3] we can observe how several techniques and architectures have been tried with some interesting results using different RF signals attributes in order to localize their targets.

2. Goals of the project

In the preceding section, we presented a comprehensive overview of various problems within wireless communications and using RF signals where machine learning has demonstrated its utility. Additionally, we provided an initial description of the problem at hand. In this subsequent section, we shall delve deeper into the problem domain, aiming to define the specific challenges, establish clear objectives for the project, and outline the corresponding specifications that will aid us in assessing the success in solving our task.

2.1 Problem definition

As mentioned in the introduction, we will perform classification of Wi-Fi transmissions sent under four different protocols. The four protocols being classified are the already explained 802.11ax, 802.11b, 802.11n and 802.11g. These are defined on the 802.11 IEEE standard.

This classification will happen only using raw IQ samples at the physical layer without any prior knowledge on the signal about where the preamble is or what information does it contain that would help identify the protocol in a normal wireless communication schema.

As part of the experimental setup, synthetic transmissions will be generated to collect a diverse dataset, simulating various real-world scenarios. To evaluate the robustness and effectiveness of the model, different levels of noise will be deliberately introduced to the collected transmissions, thereby subjecting the models to challenging and adverse conditions. This comprehensive evaluation will enable us to assess the model's performance under varying noise intensities, providing valuable insights into its resilience and generalization capabilities.

In addition to noise manipulation, we will incorporate three distinct channel models into the base signals. These channel models aim to replicate a wide range of environmental conditions that can impact the transmitter-receiver connection. By simulating different channel characteristics, such as fading, interference, and multi-path propagation, we introduce a higher level of complexity into the experimental setup. This ensures that the model is exposed to diverse and challenging transmission scenarios, allowing us to evaluate its adaptability and reliability in realistic wireless communication environments.

With the data prepared through diverse noise and channel conditions the challenge is to train a ML classifier that is able to achieve good performance in all the presented conditions. Performance will be measured through classification accuracy.

These augmentation of the dataset by applying noise and channel conditions to the signals aim to reproduce the varying environments in which ML models that are pretrained offline have to face when they are deployed in real settings. To explain this, in the next section we will go in further detail of this channel challenge for wireless transmission and the specific types of channel models that we will be applying to our IQ samples.

2.1.1 Challenges of the wireless channel

As mentioned before, the channel in which the wireless transmission takes place poses a really important problem in deploying static classifiers, as for example machine learning and deep learning based ones. Training data can follow a very different distribution than the one that will be observed when the model is performing inference in a real-life scenario.

Modeling diverse channel conditions have been an ongoing problem in the wireless and telecommunications field. Several applications have been developed with limited success in reproducing certain real-life scenarios. Some of the most important factors that affect wireless transmissions and channels itselves are the following:

- Noise: This factor corresponds to environmental noise, which affects the wireless signal by interfering with it. It can be natural or human-made, such as noise from nearby devices or radio frequency interference (RFI) from other sources. It is also known as background noise.
- Weather Conditions: Weather or climate conditions, particularly in outdoor wireless systems, can cause signal degradation. Rain, snow, fog, or even humidity can affect channel conditions by introducing signal attenuation, scattering, or absorption.
- Distance: As the distance between the transmitter and the receiver increases, the transmitted signal may be affected by attenuation, impacting signal strength and quality. This can also result in the receiver observing a more distorted version of the signal.
- Obstacles and Reflections: A wireless signal can reflect off physical obstacles. Buildings, walls, trees, or other objects can obstruct or scatter the wireless signal. These reflections can create multipath propagation, causing interference and signal degradation.
- Interference: When different wireless devices share the same frequency band, it can create disruptions in the transmitted signal. This interference can have various origins, including human-made devices or even atmospheric conditions.
- Fading: Fading refers to the variation in signal attenuation caused by changes in the wireless propagation environment. Some of the most common causes of fading are multipath propagation, obstacles, or atmospheric conditions.
- Frequency Band: The propagation characteristics are partly defined by the frequency bands being used. Higher frequency signals are more susceptible to attenuation from obstacles and atmospheric conditions.

In our case, we will not be able to modify or take into account all these elements, but we are able to modify decisive factors such as reflections, interferences, fading, noise and we work under the assumption of a fixed frequency band. These modifications in the environment give us an approximation of how our models would perform *in the wild*. We will be applying several channel conditions through the `MATLAB` engine for `Python`. To obtain the results, we use three different channel models that are explained in the following list:

(1) `Rayleigh fading [1]`: it is a statistical model employed to capture the influence of the propagation environment on a radio signal, as experienced by wireless devices. This model serves as a mathematical representation that characterizes the random variations in signal strength and quality resulting from the interaction between the transmitted signal and the surrounding multipath environment. It provides a good approximation for many real-life wireless scenarios in which multipath propagation plays a big factor in how different will be the signal received. The main assumption of this model is that the signal being transmitted will fade or vary following the Rayleigh distribution, which gives name to this channel model.

(2) `TGax`: this channel model filters a signal through an 802.11ax indoor multiple input multiple output (MIMO) channel following the approach develoved in [8]. It was implemented to test the protocol with the same postfix (802.11ax) in order to evaluate its performance of these Wi-Fi systems. Again, it will consider and model several of the mentioned factors that can affect a wireless transmission. The main aspect of this model is that it can adapt to specific deployment scenarios. In our experiments we will be using the delay profile named `Model-B`. This model sets aspects like maximum delay, breakpoint distance or RMS delay spread.It is supposed to help carrying a realistic assessment of Wi-Fi signal transmissions.

(3) `TGn [6]`: this last channel model accounts for filtering an input singal through an 802.11n mulitpath channel model. We will also be using the same delay profile. It is a very similar channel model to the TGax channel but it can only handle an smaller number of parallel transimissions and it is less efficient.

These three channel models, together with the absence of any channel model will serve as our benchmarks to assess the classifiers performance. All of them pose different challenges and environmental conditions that our models will have to face in order to achieve the desired performance. It should be noticed TGn and TGax are more similar between them and as a consequence, Rayleigh channel model is a more different channel model to these two.

2.2 Objectives

Given the problem of Wi-Fi signal protocol classification. We propose well-defined objectives that encapsulate the desired outcomes and milestones of our project. These are the following:

(I) **Train a classifier that is able to perform in several channel and noise conditions**: there is a necessity for a model that is able to generalize in the challenging conditions that we explained in the previous section. The classifier not only needs to be able to achieve good performance on the different channels, but also maintain it through different levels of noise.

(II) **Compare different neural network architectures**: we will be comparing the well established CNN in the wireless domain against the Transformer. Exploring their benefits and downsides while trying to get an overview of their internal functioning and properties that make them suitable for this problem.

(III) **Design the Transformer architecture**: in the same line as the previous objective, we will have to design the neural network through hyper-parameter tuning. The design decisions will include number of layers, learning rate, batch size and input size among other.

(IV) **Assess the performance**: we will be taking into account classification accuracy, number of parameters of the model, inference time and input size to evaluate the performance of our classifiers.

(V) **Understand and check the challenge that the channel poses**: as explained earlier, environment is able to change the distribution of data samples transmitted over the air. We want to analyze the problem and how our models are able to adapt to the changing conditions.

(VI) **Create an augmentated dataset**: in order to face the task we will have to create a dataset augmentated throught the `MATLAB` engine that shows the model the adverse conditions at training time so that the underlying protocols particularites can be learnt by the model.

(VII) Compare the change in performance when the model is shown all conditions: we will assess how performance actually changes when the model is shown specific conditions at training time.

(VIII) Provide a description of the training and testing methodologies: in order to foster reproducibility, we will explain the process we followed that lead to the presented results.

These set of objective represent the guidelines for the project and aim to present a comprehensive guide of all what is wanted to achieve in this thesis.

2.3 Specifications

In this subsection we will define a set of specifications along with some performance metrics in order to ensure a systematic and objective evaluation of the work developed in this thesis. These metrics, will serve to quantitatively measure our success and how efficient and effective our proposed solution is. We will be using metrics such as classification accuracy in several signal-to-noise ratio (SNR) scenarios or some computational efficiency indicators to evaluate and carefully assess the performance and impact of the models developed in this project. The following list will encapsulate the specifications and performance metrics we aim to achieve:

- Each transmission will only correspond to one protocol. Therefore, there will be no overlapping between the protocols being transmitted.
- This thesis will only use data generated in a synthetic manner. The reason behind this is that real-world wireless datasets are more expensive to collect and the aim of this thesis is to prove the utility of the methods described.
- The solution needs to perform well (over 95% classification accuracy) in environments with reduced signal to noise ratio, SNR over 10 dBs.
- The final architecture selected should be able to reach good levels of accuracy even in presence of high noise interference. This means an accuracy of over 80% in the range between -15 and 10 dBs.
- The proposed solution should be able to perform in a similar manner in all the environments that will be tested, these include no channel applied and Rayleigh, TGax and TGn channel models.
- Inference time for the designed models should be under 1 ms.
- Model size should not surpass the 10 million parameters, size is an important factor when working in edge devices and we should aim for a small model that is able to solve the task.
- Usage of an experiments tracking framework to save the information and results of any run during the project.

3. Proposed solution

To address the classification problem outlined in 2.1, we will employ two distinct neural network architectures as our classifiers. The first architecture employs a convolutional neural network (CNN) to leverage the extraction of local features from the IQ samples, enabling the identification of the transmitted protocol. This approach capitalizes on the CNN's ability to capture and analyze spatially close patterns within the IQ samples.

The second architecture is a Transformer-based model, which incorporates the named attention mechanism as discussed later in 3.3.1. By leveraging this attention mechanism, the model aims to derive enhanced representations of the input IQ samples, subsequently improving the classification performance. The Transformer-based architecture's ability to capture global dependencies and contextual relationships empowers the model to discern intricate patterns within the IQ samples, further enhancing the accuracy of the classification process.

It should be mentioned that the introduction of the Transformer architecture for a waveform classification problem is something somehow novel and it will be compared with the CNN. The later type of model has been more often implemented for this kind of task. The results by the CNN model will act then as the benchmark that we aim to improve.

3.1 Dataset

The data utilized for both training and testing the model consists of synthetic transmissions generated using `MATLAB`. Using this software, we produced files containing transmissions corresponding to the four protocols targeted for classification. Each file contains a continuous stream of IQ samples represented as complex numbers in the form of $I + Qj$. During the training and testing phases, these streams are divided into equally sized partitions, which serve as input data for our models. The number of generated files per protocol is summarized in Table 2.

Wi-Fi protocol	**Sampling Rate**	**MCS**	**Modulation**	**# of training files**	**# of testing files**
802.11b	20MHz*	QPSK	DSSS	2000	500
802.11g	20MHz	16-QAM	OFDM	2000	500
802.11n	20MHz	16-QAM	OFDM	2000	500
802.11ax	20MHz	16-QAM	OFDMA	2000	500

Table 2: Summary of available dataset. `*upsampled 802.11b from 11MHz to 20MHz`

It is important to note that these 2500 files per protocol will undergo random variation in terms of noise levels, ranging from -20 dB SNR to 30 dB SNR, or experience one of the aforementioned channel conditions, including Rayleigh, TGn, or TGax. Consequently, the same base signal may be utilized during training with varying levels of added noise and different channel conditions. This approach aims to augment the data and enhance the resilience of our models to different environmental conditions that make the waveform classifications problem harder. A generalized good performance of our models in different environmental settings is a fundamental objective of this project. Despite using some of the samples multiple times with different levels of noise or channel, an input base sample belonging to the train set will never be used as a test one even if a certain combination of noise-channel has not been used during the training.

Although the data is synthetically generated, the incorporation of noise and channel variations aims to create a realistic portrayal of Wi-Fi transmissions collected from real-world scenarios. Consequently, our models should demonstrate robustness in real settings without significant performance degradation. In addition, we should mention the scope of this thesis is limited to working with synthetic samples, and the use of real data will not be tested in this work and will be considered as future work.

For each protocol, in Table 2 we can observe other properties, next list aim to provide a description of this other parameters that were set during the data collection and curation stage and influence how the synthetic waveforms are created.

- `Sampling Rate`: it represents the frequency at which the analog signal being transmitted is *captured* or sampled since digital representations are always discrete. It is a concept that shows when digitizing a continuous analog signal. For the signals collected we are using a sampling rate of 20kHz, which means that 20.000 samples will be recorded per second of the given transmission.
- `MCS`: or Modulation and Coding Scheme, defines a set of of modulation schemes (such as BPSK, QPSK, 16-QAM, 64-QAM, etc.) and coding rates (such as 1/2, 3/4, etc.) that can be used to transmit data. Coding rates refer to the amount of redundancy introduced in the transmission to ensure information is accurately transmitted. Higher MCS values will provide higher data rates but can be prone to more errors in the transmission in adverse channel conditions, while lower ones provide better error resilience but lower data rates. This trade-off needs to be dynamically assessed to adjust the modulation scheme to the channel conditions. In the data that we generated, 16-QAM is the choice for every generated waveform except for 802.11b, which is not compatible and uses QPSK in this case.
- `Modulation`: This refers to the specific modulation and access technique used for the transmission. DSSS or Direct-Sequence Spread Spectrum, OFDM or Orthogonal Frequency-Division Multiplexing and OFDMA or Orthogonal Frequency-Division Multiple Access are different options available, being OFDMA the one that allows for higher data rates. DSSS is the most different modulation technique since it spreads the signal over a wide frequency band to improve resilience.

3.1.1 Signal analysis

The purpose of this subsection is to get a grasp of how the signals being received look. This will be analyzed in two different levels. First, we will be checking how different signals from different protocols look between each other. Secondly, we will look at how much does the environment impacts the transmitted waveform.

To begin with, Figure 2 provides a glimpse of an example window of a waveform for each of the four protocols under consideration. As mentioned previously, the IQ samples are represented and saved as complex numbers, and the plots depict the magnitude of these complex numbers. Consequently, the dynamic range for all the plots does not fall below zero. Upon examination, it becomes very apparent that protocol 802.11b exhibits the most distinct characteristics compared to the others. This discrepancy can be attributed to the distinct modulation technique employed by this particular protocol, which differs significantly from the rest. Also it is worth mentioning that the signals for 802.11b protocol are not always one, but due to the modulation technique that it uses the magnitude of them are. Another noteworthy

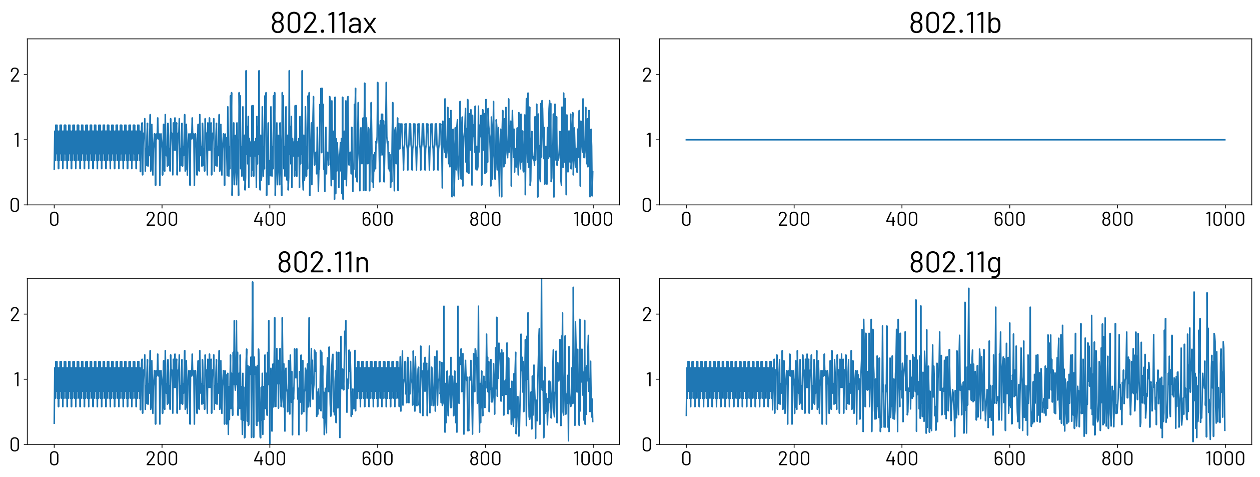


Figure 2: Example window display for each of the protocols, horizontal axis is for time or index of sample and vertical axis is the module of the complex representation of the I and Q component.

observation from this figure is the striking similarity between the remaining protocols. Both 802.11n and 802.11g showcase nearly identical dynamic ranges, while 802.11ax exhibits a slightly lower dynamic range. This initial analysis enables us to assess the appearance of received signals when no channel effects or noise are present. Furthermore, it serves as a valuable reference point for evaluating subsequent signal variations introduced by channel conditions and noise.

Now, let's delve into the impact of noise using Figure 3. This insightful figure showcases six distinct levels of noise superimposed on the same 802.11ax base waveform, which is a waveform present in our training data. As a reminder, in the experiments the intention is to simulate a continuous range of signal-to-noise ratio (SNR) values spanning from -20 dBs to 30 dBs by applying varying levels of noise. By closely examining the figure, we can observe two noteworthy phenomena arising from the introduction of noise.

Firstly, as the noise intensity increases (or SNR decreases), the dynamic range of the signal expands. This can be attributed to the fact that in order to decrease the SNR, which represents the ratio of signal power to noise power, if the signal power is fixed (it is always the same waveform) we can only increase the power of the noise component. Consequently, the overall power of the received signal also increases.

Secondly, the shape of the signal undergoes notable changes in the presence of noise. This can be attributed to the inherent nature of Gaussian noise, which is the type we are using and is randomly distributed following the Gaussian distribution. As a result, the noise introduces a non-deterministic variation in our signal, giving rise to distinct waveform patterns even when the base signal and noise level are the same.

These SNR levels indicate that we will have training and test samples where the signal power is 1000 times stronger than the noise power, corresponding to 30 dBs. Additionally, we will have samples where the waveform strength is only one-hundredth of the noise power, corresponding to -20 dBs. By training our models with this wide range of SNR values, we aim to develop robust and resilient signal processing algorithms capable of handling real-world scenarios where noise plays a significant role. This variation in SNR levels ensures that our models can effectively handle the challenges posed by different noise intensities, ultimately enhancing their performance and adaptability in practical applications.

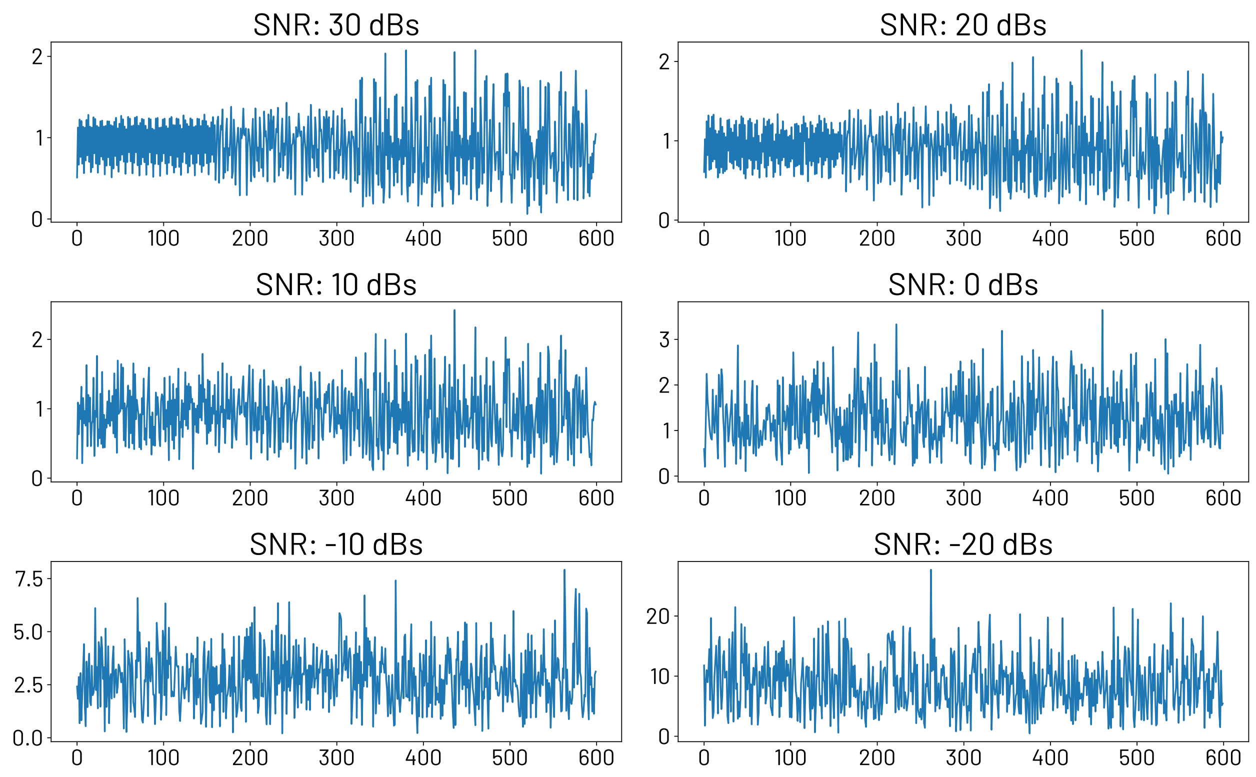


Figure 3: Evolution of how different levels of noise affect the same signal received. In this case the base signal in which the noise is applied is the same for each frame and belonging to protocol 802.11ax. The models will be presented samples with noise down to -20 dBs. Horizontal axis is for time or index of sample and vertical axis is the module of the complex representation of the I and Q component.

In addition, Figure 4 provides us with insights into how different channel conditions impact Wi-Fi transmissions. To demonstrate this, we passed a 802.11b waveform through various channel conditions to visualize their effects. It is evident that the environmental conditions we simulated have a significant influence on the dynamic range and shape of the received signals.

In particular, the TGn and TGax channel conditions substantially reduce the dynamic range, resulting in a magnitude reduction on the order of $1e-3$. On the other hand, the Rayleigh fading channel does not have as pronounced an effect on the range of received values, but it does introduce considerable variations in the shape of the transmitted waveform. This is noteworthy as the base signal was originally a constant magnitude transmission.

The Rayleigh fading channel exhibits also the largest difference between the maximum and minimum values, further emphasizing its impact on the transmitted waveform. It is important to highlight that each of the channel models significantly alters the received signal, leading to variations in the waveform characteristics.

After analyzing and evaluating the visual representations presented in Figures 3 and 4, it becomes evident that the influence of the environment, whether it manifests as a channel model or as Gaussian noise, exerts a significant and profound impact on the wireless transmission. This realization highlight the necessity of augmenting the dataset to facilitate the development of robust and adaptable models

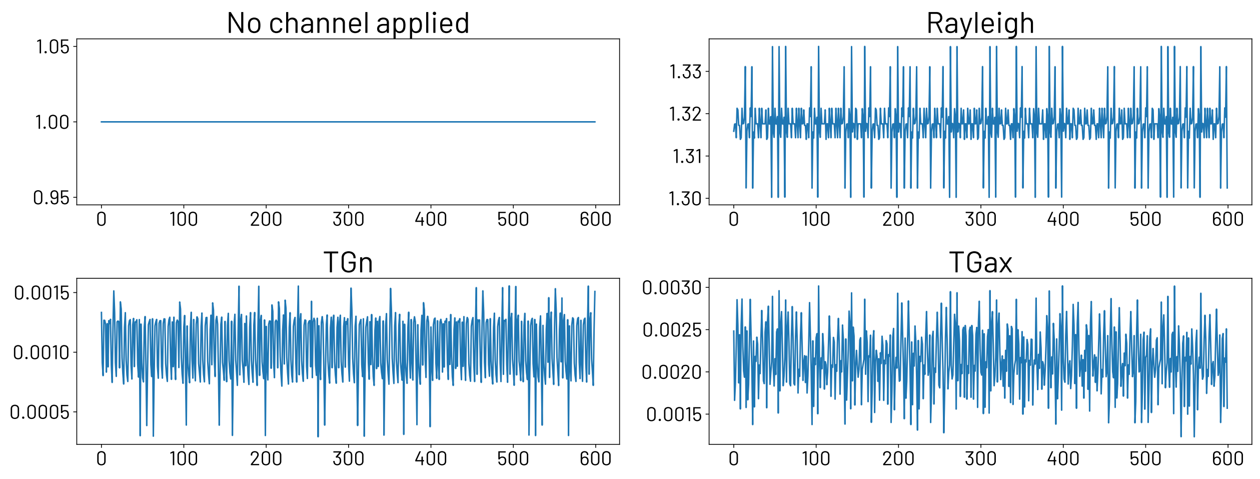


Figure 4: Visualization of how the different channel models that we will apply modify the signal. Notice the change in dynamic range for each of the channels. Horizontal axis is for time or index of sample and vertical axis is the module of the complex representation of the I and Q component.

capable of effectively generalizing and performing well in unforeseen and diverse environmental conditions. By augmenting the dataset, we can enhance the capacity of our models to encapsulate the intricacies and complexities inherent in real-world scenarios, thereby fostering a more reliable classifier in this kind of deployments.

Lastly, it should be mentioned that the loading and saving of the signals once are generated are through the implementation of a `Pytorch` dataset class. Thanks to this class, we can easily load the signals as batches using the integrated `Pytorch` dataloader object.

3.2 CNN

The initial classifier proposed as a potential solution consists of a combination of a convolutional neural network (CNN) followed by a multi-layer perceptron (MLP) head. CNNs offer compelling advantages in this context, such as their ability to exhibit spatial invariance, which allows for effective pattern extraction across different regions of the input. Additionally, CNNs leverage a relatively smaller number of parameters compared to other architectures, leading to more efficient and manageable models. Moreover, the inherent design of CNNs facilitates the direct input of raw data, as the neural network simultaneously serves as a feature extractor. The convolutional block will serve as the backbone to extract information and representations from the raw IQ samples that are the input to the model.

This type of neural network architecture has been around since the 1980s and it has served as a baseline deep learning framework for several works in the wireless communications domain as analyzed in Table 1. The already discussed advantages that this architecture bring to the table made it very suitable and the first choice in several problems. That is why the performance of this classifier will act as a benchmark that the Transformer-based models will have to try to improve.

Our classifier model (Fig. 5) is composed of a layer norm in the temporal dimension or S. This layer is followed by two convolutional layers with a Max pooling in the S dimension and then by two fully-connected layers. The input is a tensor or slice of size $2x512$ containing the windowed IQ samples. To create the

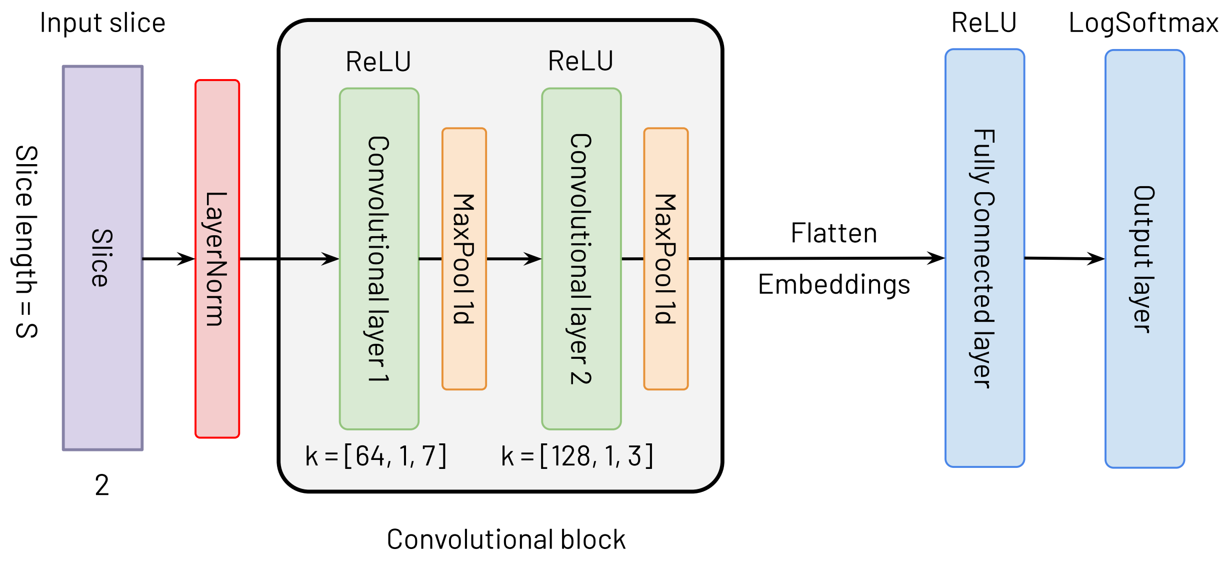


Figure 5: Chosen convolutional neural network architecture with a Feed Forward classification head.

input of this model we will just partition the signal in the file creating the 2*xS* slices. One dimension of the input is the in-phase samples and the other one is the quadrature-phase samples of the waveform. The first step, a normalization of the input I and Q dimensions is intended to improve the performance of the model against the different dynamic ranges that we saw in 3.1.1. This tensor goes through a first convolutional layer consisting of 64 filters or *kernels* of size 1*x*7. We can interpret this step as extracting features on each of the real and complex parts of the signal separately. Afterward, the result of this first layer goes through a Max-Pooling layer also individually in the 2 dimensions. Following this, the output is processed by a second layer with 128 filters of size 1*x*3 with another Max-Pooling operation after it. In this second one, we also try to extract information from both phases of the signal separately. The output of this layer is then flattened and fed into a first Fully Connected layer with 256 neurons. Finally, the resultant tensor is connected to an output layer with a number of output values equal to the number of protocols. This last layer is followed by a *logSoftmax* function that will return the logarithm of the corresponding probability distribution over the protocols that the Softmax outputs. The two convolutional layers and the Fully Connected one after the convolutional block are followed by a Rectified Linear Unit (ReLU) activation function acting as a non-linearity for our model.

The result is a vector of 4 positions where each position represents the logarithm of the probability that a certain protocol is used as the standard for that specific Wi-Fi transmission. We obtain the classification accuracy by comparing it to the ground truth of the input vector. The ground truth is available since we know to which recording those values belong, and the recordings are saved in folders depending on which protocol they belong to. This metric will determine the quality of the classification and allow us to compare our results with the ones obtained by the models proposed in 3.3.4.

3.3 Transformer

The second architecture we will explore for the problem is Transformer-based. This architecture was introduced in the highly cited paper "Attention is all you need" by A. Vaswani *et al.* [14]. It is a neural network architecture widely used in the recent years and proven effective in several domains such as natural language processing (NLP), computer vision, speech recognition and synthesis or recommendation systems. Transformer-based models have revolutionized applications such as machine translation, text generation, sentiment analysis, text-to-speech or object detection among others.

Although it is true it is also suitable for applications in which non-sequential data is used. It has been mainly applied with sequential data. The reason behind it is that the data that is inputted to this architecture needs to be represented as a sequence of tokens. Each token represents a unit of information. For example, in the context of NLP tokens can be words (subwords or even characters). Any of these, represent a smaller division of the whole sequence which in this case would correspond to a sentence.

The key aspect of this architecture is the so called attention mechanism. It allows the model to create better representations of the inputted tokens by mixing their information and allowing each of the tokens to *pick* information from the other tokens present in the sequence. The way in which this is done will be explained more in detail in 3.3.1. The contextualized embeddings or representations after the Transformer encoder layers will be followed by a Fully Connected layer and an output layer with a neuron for each of the four classes in our problem. Since the task for our model is simply a classification one, we will only be using the encoder part of the full Transformer architecture. This is due to the fact that we are not generating any sequence when predicting which protocol a transmission belongs to.

3.3.1 The attention mechanism

As mentioned earlier, the attention mechanism is the key piece inside the Transformer architecture. It allows each of the tokens within a sequence to attend to the most relevant parts of the same or different sequence. This enables the tokens to utilize their token embeddings to create an improved representation that is contextualized among the accompanying tokens and enhances the network's ability to learn. The way in which this architecture achieves this is similar to the schema used in recommendation systems. In such systems, you typically compare the similarity of a given query to some keys or information that describe the actual content you want to retrieve. Depending on the similarity or the quality of the match, you return the most relevant result first.

In the Transformer context, each token of the input sequence, after going through a linear transformation, will be transformed into a query, a key and a value vector. Then by doing the cross product between the query of a token with all the token's keys we will obtain the attention scores. These values will be normalized and applied a Softmax. Finally, the embedding that will come out of the attention operation is the weighted sum of the values of the tokens, with those attention scores or query-key similarities computed acting as the weights. We just described the attention formula (equation 1) as it is in [14]. This formula gives us a description of how the attention mechanism uses Q, K and V, which are are the queries, keys and values matrices respectively and $\sqrt{d_k}$ is a normalizing factor representing the square root of the dimension of the key vectors.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

Another important piece to explain and pay attention to is the multi-head attention. The concept behind it is that we will now give the model the ability to attend at different pieces of information of the input in order to produce even better representations for the tokens. Equation 2 describes the formula used to compute the multi-head attention.

$$\begin{aligned} &MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \\ &\text{where} \\ &head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2}$$

These two concepts are the base of the Transformer architecture and started a revolution and a shift towards this kind of neural network in the field of artificial intelligence. Some of the other advantages it provided was eliminating the recurrence mechanism inside recurrent neural networks, which were previously the most used type of architecture to work with sequential data. This means that it can process the whole input in a parallel manner since it does not require the computation on a previous token to compute the representation of another one. The other advantage is that with this purely attention-based design we allow the model to access all other tokens at any given time directly to bypass any kind of information bottleneck in the sequence.

3.3.2 From IQ samples to tokens

Bringing this to our current problem, we are presented with an stream of continuous IQ samples. This is our sequential data. The first step is dividing this uninterrupted flow of complex numbers in vectors of size $2xN$. By doing so we will obtain the sequences that serve as the model input. As we also need the tokens that form the sequence, the approach we take is splitting this N samples conforming the sequence into equally sized chunks of S IQ samples. That will correspond to our *words* inside the *sentence*. The way in which we do that can be visualized in Figure 6.

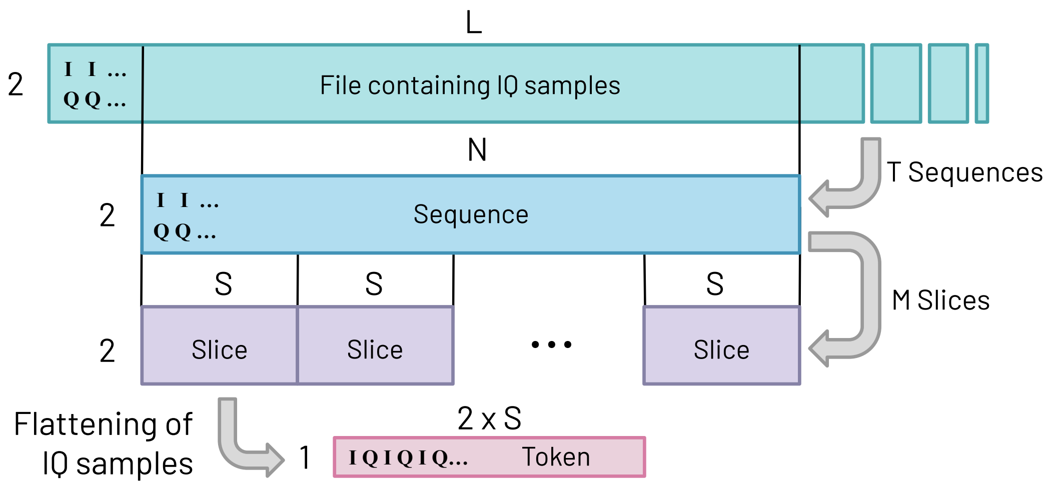


Figure 6: Partition of transmission file into sequences, sequences into slices and slices into tokens by interleaving the IQ samples.

An additional crucial aspect needs to be addressed. IQ samples represent a 2D stream of data, as they encompass both the in-phase and quadrature-phase dimensions. However, our tokens should consist of

1-dimensional vectors. To overcome this challenge, we adopt a strategy of interleaving the values from the I dimension with those from the Q dimension, thereby creating tokens that serve as input for the model. In essence, we merge the two dimensions by zipping them together, resulting in a single vector with twice the length. This process can also be visually depicted within the same figure, providing a clearer understanding of the transformation.

Now, each of the M sliced vectors or tokens inside the continuous stream of IQ samples will go into the Transformer model layers to get an improved representation of the input of the model that will help enhance the performance of our model.

3.3.3 Hyper-parameter tuning

To choose the hyper-parameters we use for the training and design of the Transformer-based model we conducted a sweep. A sweep consist of using different combinations of hyper-parameters for a model trying to maximize a metric (validation accuracy in our case). We also do that trying to use the smallest model possible in terms of number of parameters. Figure 7 shows all the combinations that had been tried during the hyper-parameter tuning phase. Each line indicates what value was chosen for each hyper-parameter for a particular run and the color gradient the value of validation accuracy that was obtained for that specific combination. Notice that all combinations obtained high accuracy values, higher than 96.5%, but we aim at getting the maximum accuracy possible.

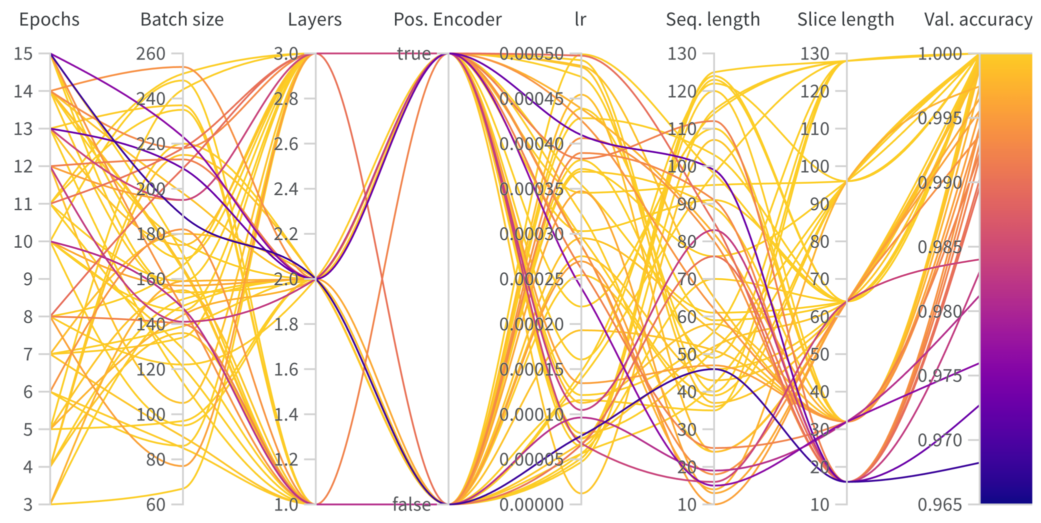


Figure 7: Parallel coordinates plot for hyper-parameters sweep.

After analyzing the results from the sweep, we decided to use as final hyper-parameters the following values. For number of epochs it seems that training for longer time is not beneficial to the final result so we choose 5 epochs as our training time. A batch size of 122 and learning rate of $lr = 0.0002$ were chosen although there is not a strong pattern in the performance when varying these values in the designed ranges for the experiment. To give the model enough capacity to obtain great performance, we also choose 2 layers.

For the use of positional encoding, the decision taken is not to use it. The sweep indicates that the model performs as well or better without it. It makes sense not to use a positional encoding since our tokens have really low probability to be repeated in the presence of noise or channel, so every *word* that the model sees is different, although all of them are similar. This is not the case in classic NLP where the vocabulary or different number of tokens is closed, known and discrete. In this way, we also avoid the overhead of the operation at inference time.

Finally, for sequence length (M) and slice length (S) we choose two different combinations. The justification for this decision is that using these two configurations will allow us to compare two models with a different number of parameters. The first combination is $M = 64$ and $S = 128$. This will determine the input size for the named *large* or *LG* architecture. The second one, smaller in number of parameters, will have $M = 24$ and $S = 64$. The name for this architecture is *small* or *SM*. These choices for sequence length and slice length will let us explore the trade-off between having a smaller model that should do inference faster and a larger one that should be able to get better test accuracy as it has a bigger number of neurons and learning capacity.

3.3.4 Final architecture

After the hypertuning step we have two final designs for our classification task using the Transformer encoder block. Both of them uses the same number of Transformer encoder layers. The difference happens only in the size of the sequence and the tokens as explained in the previous section. As a consequence for this difference in input size, the LG model will end up having more than 5.2M parameters more than the SM one.

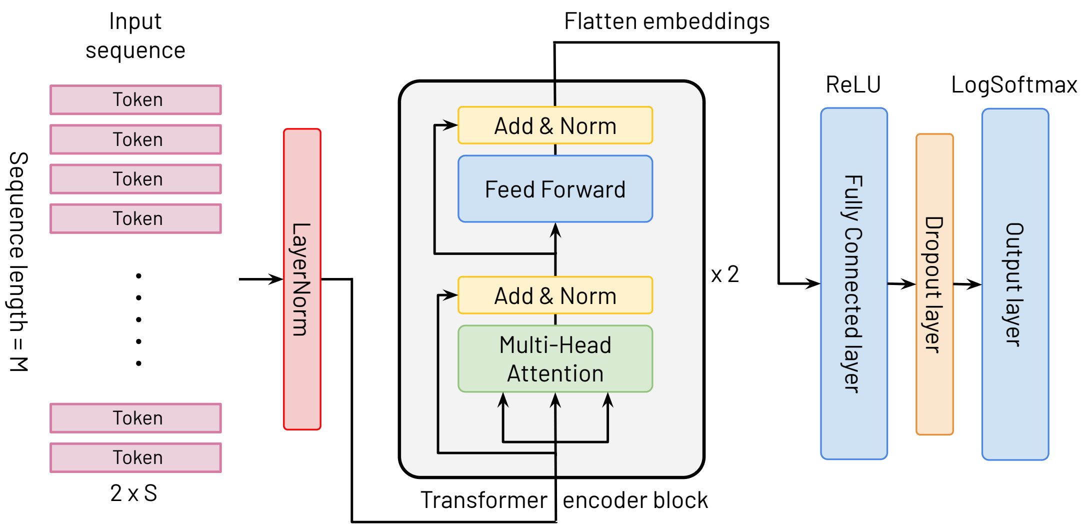


Figure 8: Chosen Transformer-based model architecture. Parameters M, S of the input size will be different for the two model sizes. For the small architecture (SM), M=24, S=64 and for the large one (LG) M=64 and S=128.

Figure 8 presents the architectural blueprint of our Transformer-based neural networks. The input comprises a sequential arrangement of contiguous tokens, which undergoes an initial processing in the token or $2xS$ dimension through a normalization layer. The reason for this layer is the same as it was in the CNN model, the change in the dynamic range when a signal goes through channel or noise forces us to add this step. Following this, the normalized IQ samples vectors enter two Transformer encoder layers. Within each layer, the tokens undergo multi-head self-attention, accompanied by the inclusion of a residual connection originating from the pre-attention stage and a subsequent normalization layer. Additionally, the encoder features a Feed Forward layer, followed by another residual connection and subsequent normalization. This entire process is repeated twice, after which the resulting embeddings from the Transformer encoder block are flattened. After this, the output of this block proceeds through a Fully Connected layer with a neuron count of $2xS$, activated by a ReLU function. Here, S corresponds to 128 for the LG model and 64 for the SM model. Lastly, a dropout layer with a probability of 0.5 is applied, followed by a final output layer and LogSoftmax activation function. This arrangement generates logarithmic probabilities indicating the likelihood of the input transmission belonging to a specific protocol.

3.4 Training & Testing

For each architecture selected, namely the CNN-based and the SM and LG Transformer-based architectures, we will train models in three different settings. For first one, models will be trained in static conditions of noise (with chosen 10 dBs of SNR) and channel. This means that for this first experiment we will have 12 different models which correspond to the three architectures selected, in each of the four channels available (counting the absence of one as a possibility) with fixed 10 dBs SNR for all of these trainings.

The second setting will consist on keeping a fixed channel during training time, but augmenting the IQ samples with variable levels of noise from -20 dBs to 30 dBs SNR. The specific choice for a training sample will be picked from a uniform distribution with the SNR limits as the continuous range of options to pick from. So every time a sample is retrieved from the dataset, a random noise level between those two values of SNR will be applied to the signal before entering the model. Notice that as mentioned before, -20 dBs of SNR means that the actual ratio of signal strength to noise strength in this case is 0.01, which means the signal power is one-hundredth of the noise power. And the other extreme, 30 dBs, represents that the actual ratio of signal strength to noise strength in this case is 1000, indicating that the signal power is 1000 times stronger than the noise power. So this two scenarios are very different, being the negative side a harder setting for the model. Also it should be reminded again that the noise applied follows a Gaussian distribution.

The result of this second experiment will be again 12 trained models, which correspond to the three architectures trained for each of the specific channels. The difference is that this time the noise will not be fixed but variable within the mentioned range. It should be remarked how this noise is applied dynamically to the base signals already generated through `MATLAB`.

Finally the third experiment consists on presenting the three different types of models with samples from all the channels and variable noise level during training time. This means that we will have a CNN-based model, a SM and LG Transformer-based model that will be trained with samples with no channel applied, Rayleigh, TGn and TGax. In addition, a variable value for the noise also will be applied in the same manner it was applied in the second setting. These models are expected to generalize better since they have been trained in the most different number of scenarios.

These three different settings or experiments will allow us to explore how the different architectures compare between themselves while also assess and evaluate the importance of presenting the model with as much variety as possible in terms of noise and channel conditions. The reason for this is to obtain generalized good performance in all the different environments and even new ones that the model has not seen during training.

Getting now into some technical details for the training that we implemented, for the CNN we have used a batch size of 512 tensors of size 2*x*512. Adam optimizer with $lr = 0.001$ and 5 epochs as training time. For the Transformer-based models we use a batch size of 122 tensor of size 64*x*256 ($Mx(2xS)$) for the large architecture and 24*x*128 in the case of the small one. Also Adam optimizer with 5 epochs of training time but a $lr = 0.0002$ as decided in the hypertuning phase. Finally, for all the architectures the loss function we will use to optimize the model is the negative log likelihood loss or NLLLoss since it is useful to train a classification problem with C classes.

During training, 20% of the available base Wi-Fi transmissions will be saved for validation purposes. This validation samples will also be augmented through channel and noise so that the validation accuracy is a real measure of the overall performance of the model.

The only remaining part of our ML pipeline is testing. For this purpose, we collected a completely new set of transmissions that were augmented through each of the available channels. This means that for each collected transmission file of a specific protocol, we will have four versions of the same signal. One without any channel conditions applied (the base waveform), and three corresponding to each specific channel condition available. As a result, the size of the testing dataset is four times the original size. Unlike during training, where channel environments are applied dynamically, for testing, the channel environments are applied statically. This ensures that the test conditions are equal or at least very similar for all the models. Therefore, it creates a fair testing environment for the three models.

During the testing phase, we will incorporate noise levels dynamically into the evaluation process. This approach is chosen to overcome the challenges associated with managing a vast number of files if we were to save a separate file for each channel and noise level under consideration. By setting a fixed random seed in our `Python` script, we also ensure consistent generation of noise across multiple model tests. This strategy allows us to assess the performance of our models in diverse and noisy environments.

To comprehensively evaluate the performance of our models in terms of classification accuracy, we will systematically apply predetermined SNR levels ranging from -30 dBs to 30 dBs, with increments of 5 dBs. By analyzing the accuracy achieved at each selected noise level and channel condition, we gain valuable insights into the model's performance under specific environment conditions. This meticulous evaluation process enables us to assess the models' robustness and reliability across varying noise levels and environmental conditions and aids in making informed decisions about the suitability of a particular training setting and architecture for a given deployment.

3.5 Models size comparison

This final subsection of the proposed solution block will give a brief overview on the sizes of the three different models that we will compare. For instance, Figure 9 portrays a visual representation of the information that we want to convey. There is clearly one model design dominating in both categories which is the Transformer-based large architecture. It has 6.8 M parameters and the total number of IQ samples needed for an input sequence is 8192, which is the result of having 64 tokens of $S = 128$. The second biggest model in number of parameters, with 4.1 M, is the CNN-based. The convolutional block has an elevated number of kernels which produces that when the flattening step is performed, the number of parameters increases in a great manner. Despite this, it is also the model that needs the fewest amount of IQ samples to predict, with 512 IQ samples. Finally, the SM Transformer is the model with the least number of parameters, 1.6 M, and the second in IQ samples per input at 1536. The LG model uses 16 times the amount of IQ samples per input that the CNN uses, while the SM uses only 3 times the number of IQ samples.

In both of the metrics presented to assess model size we aim at minimizing them. The number of

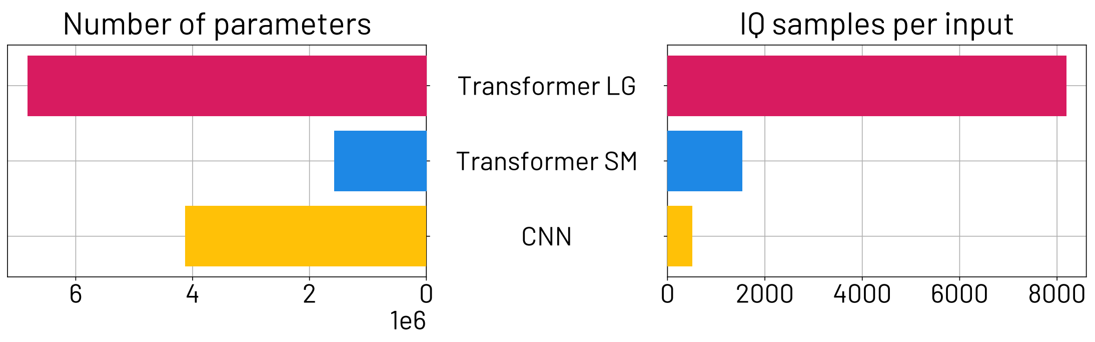


Figure 9: Comparison in model sizes both in terms of number of parameters of each architecture and number of IQ samples needed to do inference over the input.

parameters is a very important factor in wireless deployments since edge devices normally do not count with specific hardware to store and load ML models. So it is very typical to find a setting in which memory and computing power is scarce. The biggest the model is, the more memory it needs and the more computational resources will be needed to perform inference.

The number of IQ samples needed to perform inference is also of high importance since it will limit the amount of predictions we will be able to make in real time. Ignoring the computational time, if we need to wait to receive 8192 IQ samples for a single prediction it will take longer than if we need to wait for only 512. The difference, since we are talking of a wireless transmission, is of the order of ms but it is still important to keep in mind and optimize. With a model that needs a bigger input we will take longer to observe a change in the protocol being transmitted just by the fact that we need to wait longer to have a complete input. But also, with a bigger input is more probable that we make more accurate predictions, so there is clearly a trade-off that needs to be addressed.

In this comparison the LG Transformer-based architecture is clearly the biggest one. So just in terms of model size, we would prefer any of the other two models. It needs to be taken into account that model size is only one of the factors that are important to analyze when selecting a model. In the results section (4) we will evaluate also inference speed and probably the most determining one when choosing a model in any problem, that is how well does the model perform for the task that it has been designed for. This corresponds to classification accuracy in our particular case. And it is common that the correlation between model size and overall performance is high.

4. Results

In this section, we will present the results of the three experiments described in 3.4, providing a comprehensive evaluation of the performance achieved by the proposed models in those settings. Furthermore, we will conduct a global comparison of the speed and performance of these different architectures that we use. To ensure a comprehensive assessment, we will also consider the size comparison discussed in 3.5.

To provide a recap of the upcoming results, we will be showcasing the outcomes of our comparative analysis involving three architectures: CNN, LG Transformer, and SM Transformer. Each architecture will be evaluated across three sets of trained models, corresponding to specific experiments. Let's delve into a summary of the objectives and explanations for each experiment:

(I) **First experiment**: In this initial experiment, we trained each architecture under a fixed noise and channel condition. Specifically, we selected a noise level corresponding to an SNR of 10 dBs. For each of the three neural networks, we generated separate models trained for this constant noise level and each channel condition, producing a total of four models for each architecture.

(II) **Second experiment**: The second experiment focuses on training models for a specific channel condition while varying the noise level. We introduced a range of noise levels, spanning from -20 dBs to 30 dBs. Similar to the first experiment, we generated specific models for each channel condition; however, this time, the models were trained using samples containing varying levels of noise.

(III) **Third experiment**: In the final experiment, we aimed to create models that encompass a broader range of scenarios. We trained three different models, one for each architecture, using samples from every channel condition and varying noise levels within the same SNR range as the second experiment.

In total, we have trained 27 models: 12 models resulting from the first experiment (obtained by multiplying the number of architectures by the number of channel conditions), 12 models from the second experiment, and 3 models from the third experiment. The primary objective of these experiments is to observe the performance variations across different settings and assess the effectiveness of each classifier type. By analyzing the results, we aim to demonstrate the significance of collecting and showcasing data from diverse scenarios to achieve robust and generalizable models.

Through this comprehensive evaluation, we expect to gain insights into the strengths and limitations of each architecture, identify optimal settings for specific conditions, and determine the most suitable classifier for this particular task. The results will also highlight the importance of training models across a range of scenarios.

To ensure efficient and organized training, we leveraged the powerful experiment tracking framework called `Weights & Biases`. This framework allowed us to record and store detailed information about each training run. By utilizing the intuitive Workspace provided by `Weights & Biases`, we gained access to a range of valuable representations and graphs that facilitated the evaluation of our models based on the chosen metric, which, in our case, was validation accuracy.

This remarkable tool provided us with an easy way to assess the success of our training runs and save the obtained results. It eliminated the need to worry about losing any valuable data or manually recording important metrics. With `Weights & Biases`, we could effortlessly track the progress of our models throughout the training process and analyze the performance achieved.

During the testing phase, where the actual results are obtained, we ensured that the outcomes were appropriately saved for further analysis. Specifically, a dedicated file was created to store the results for each noise level and channel condition, corresponding to each individual model. This allowed us to conveniently process and analyze the collected data, ensuring accurate reporting of our findings.

4.1 First experiment

To present the results of the first experiment, we will first examine the performance of one architecture trained for each specific condition, consisting of a particular channel condition and a noise level equivalent to 10 dBs SNR. We will evaluate how these trained models perform when applied to different channel conditions. Additionally, we will provide the averaged performance across all channels for the three types of models in this first experiment.

In order to obtain the averaged results, we calculated the average test accuracy for each noise level across the four different channel environments. This approach allows us to assess the overall performance of the models and observe how they handle variations in channel conditions.

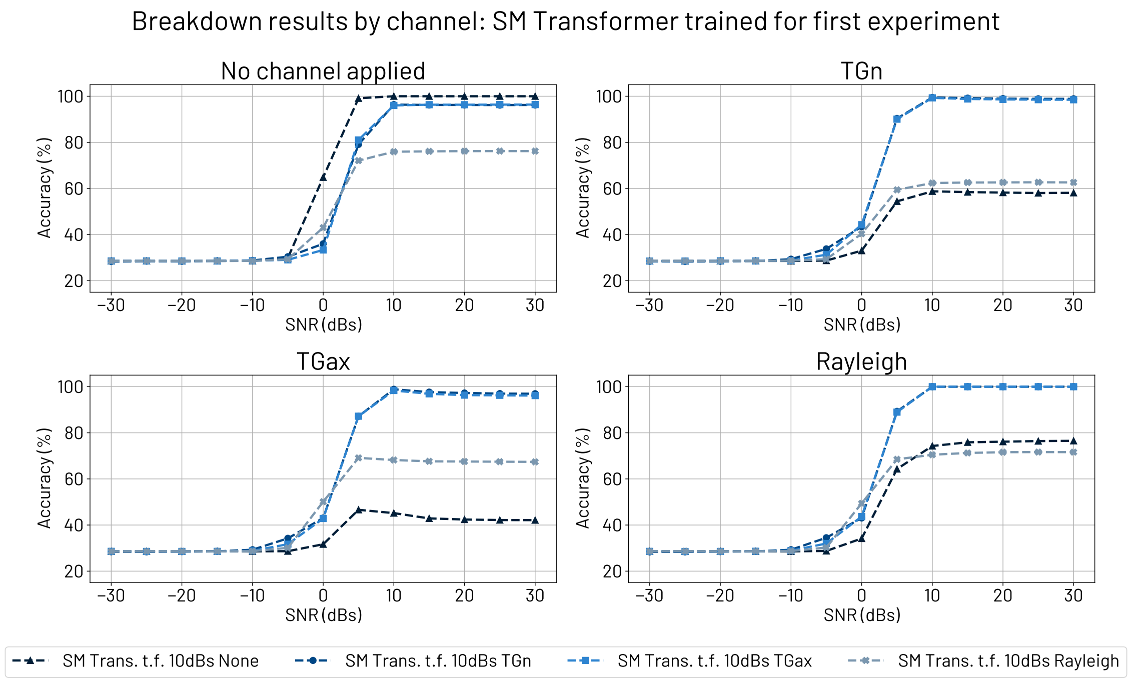


Figure 10: Comparison between SM Transformer-based architecture trained for specific channel and noise conditions. Performance breakdown by channel condition. `t.f. stands for trained for`

Figure 10 shows us for the SM Transformer-based architecture how performance in terms of classification accuracy varies when a given model is trained for very specific conditions and then used in very different ones. We have the four SM models, each trained in a specific channel condition. Except for the model trained for Rayleigh fading channel, in all the other cases the architecture trained for that channel is the one that seems to classify the best. It should be pointed out that the models trained for TGn and TGax channels seem to be highly accurate even in conditions that they have not seen during training. It is also worth noticing that performance is peaking at 10 dBs SNR, which is the noise level in which the models have been trained on, even if higher SNR should be easier as it means that the noise level is lower.

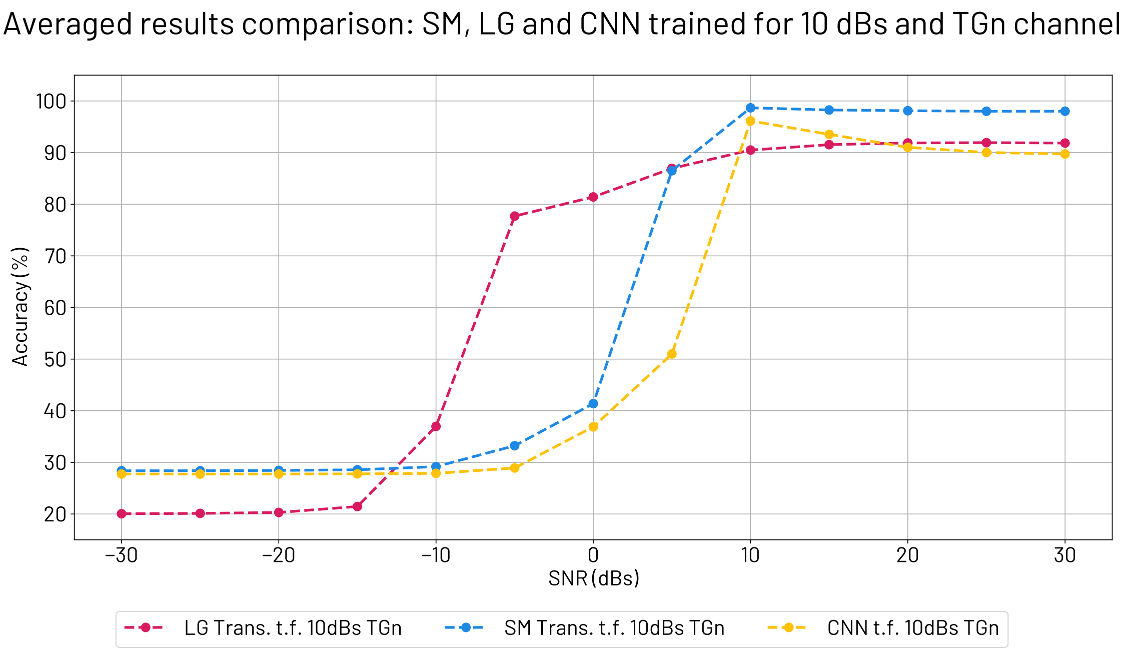


Figure 11: Average performance for the three NN types considered in the context of the first experiment. Models presented in this case were trained for TGn channel.

The second graph in this subsection, Figure 11 represents the averaged performance of the different type of classifiers that we are training and comparing in this thesis. The channel in which these models were trained is TGn, because as we saw in the previous figure, it seems that showing the models this channel makes them generalize better. What we can observe is that the LG Transformer-based model is able to maintain performance even in lower SNR levels, which indicate a higher presence of noise. Despite that, it does not perform as well as the SM model. Analyzing the individual channel results, we can find a reason for that. The large architecture seems to be overfitting in some of the higher noise scenarios, making this model not work as well in the other high noise channel conditions. CNN get similar accuracy as LG Transformer-based but it is the model which its performance drops the quickest when noise increases. Finally, it is worth mentioning that in this scenario in which we are not presented with a lot of data variety, the SM model seems to be the best option, although the LG Transformer-based is showing better endurance against noisy situations.

These results allow us to check what is the effect of presenting the model with very specific conditions during training time. As we have been able to observe, the LG architecture happens to overfit in this scenario since it is too large in terms of parameters, and tends to behave in this way when it is not trained in sufficiently diverse environments.

4.2 Second experiment

In this second experiment, we will begin by assessing the performance of a single architecture trained across different channels under varying SNR values. This evaluation will be represented in a graph similar to the one shown in Figure 10. However, in this case, the results will be specific to the second experiment settings and will focus on the LG Transformer-based model.

Additionally, we will present another plot in this subsection that enables us to compare and analyze the averaged results of the three architectures trained for a specific channel condition. In this case, we have selected the TGax channel condition for evaluation. This choice is based on the observation that TGax, along with TGn, has demonstrated better generalization capabilities, making it a suitable environment for comparative analysis between the architectures.

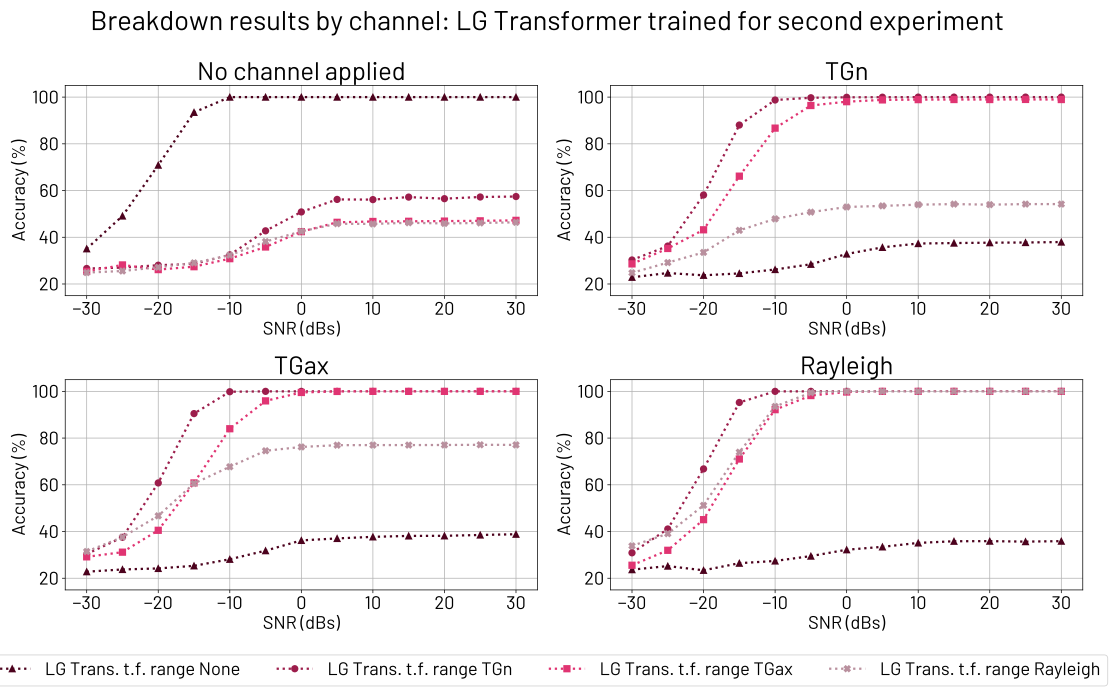


Figure 12: Channel performance breakdown for LG architecture trained for each of the channels and SNR ranging from -20 dBs to 30 dBs.

Analyzing Figure 12 in depth, we can make several interesting observations. Firstly, it is evident that the LG model trained specifically for the no channel condition applied struggles to achieve satisfactory performance in any of the other channel conditions. Similarly, the other three models also fail to achieve classification accuracy above 60% for the evaluated noise levels when being tested in the absence of a channel environment. This phenomenon can be attributed to the significant impact of channel environments on the shape of the base signals. Not applying any channel, surprisingly, emerges as the most distinct environment among those being evaluated.

Among all other channel conditions, the model trained for TGn demonstrates the best performance. It consistently exhibits higher classification accuracy levels for high signal-to-noise ratio (SNR) and maintains relatively stable performance even as the noise level increases. Notably, the TGn model outperforms the other models even in conditions for which they were specifically trained. For instance, when tested in the TGax conditions, the TGn model surpasses the model trained for TGax. Similarly, when evaluated in the Rayleigh fading channel, the TGn model outperforms the model trained for that particular channel.

These findings highlight the robustness and adaptability of the TGn model, as it consistently performs well across various channel conditions. It suggests that the TGn model has learned features and patterns that are effective for classification across different environments. On the other hand, the relatively lower performance of the other models in their respective trained conditions indicates the need for further exploration and improvement.

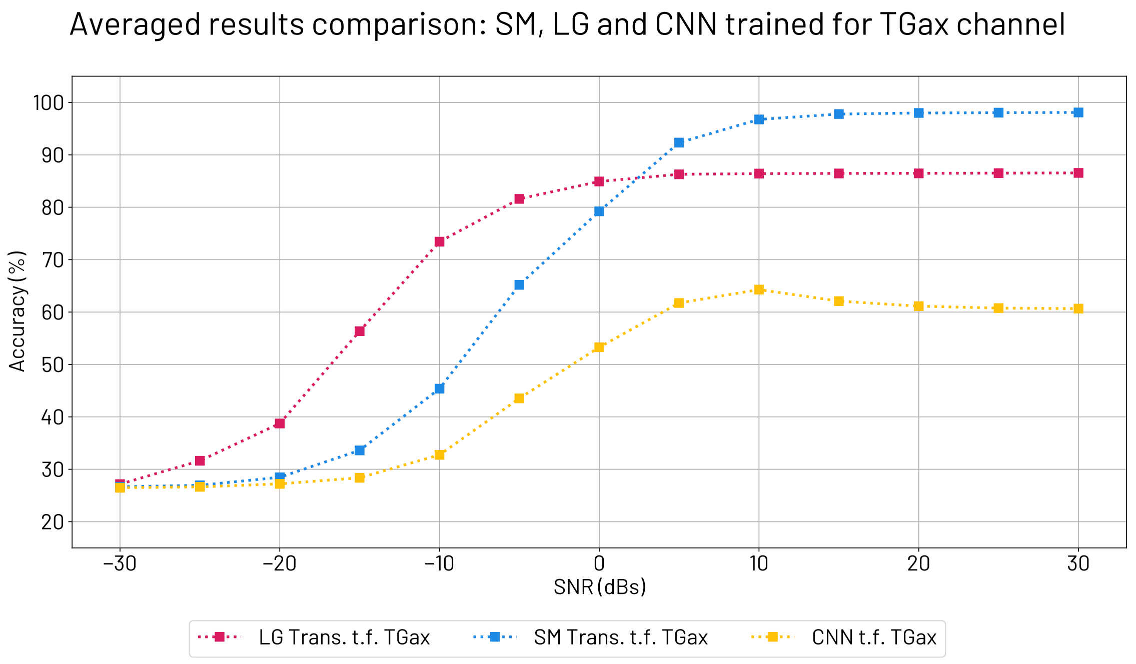


Figure 13: Comparison of average performance of the three classifiers trained for TGax channel condition and SNR in the range -20 to 30 dBs.

Now, let's analyze Figure 13. The first notable observation is that once again, the LG Transformer architecture demonstrates a higher resilience to noise compared to the SM Transformer-based model and the CNN-based architecture. This consistent pattern aligns with our findings from the previous experiment. However, it's important to note that the LG architecture still struggles to achieve optimal performance in the absence of any channel effects, resulting in a deviation from near 100% accuracy in the average measurement. As it also happened before, this does not occur to the SM model, which is able to achieve nearly perfect classification accuracy with high SNR.

Another interesting observation emerges when comparing the CNN line in this figure with the corresponding line in Figure 11. For the CNN model, training it under specific conditions of noise and channel appears to help improve its accuracy values. However, it is still not able to compete on par with the other two models in either of the settings. For this second experiment, we can observe that the CNN architecture is more susceptible to the impact of noise levels, and its accuracy peaks at around 65%.

Furthermore, when comparing the results of this experiment to those of the first experiment, it is evident that the models are now capable of maintaining performance even under adverse noise conditions. By training the classifiers with samples that exhibit similar noise levels to those encountered during testing, the models have shown increased resilience in low SNR conditions.

4.3 Third experiment

In this final experiment, our objective is to evaluate the performance of our classifiers when they have been trained on all possible conditions encompassing different channel environments and noise levels. We anticipate that this comprehensive training approach will lead to increased accuracy and demonstrate the generalization capabilities of our models. To assess this, we will compare the performance of the three models for each channel condition and SNR level under evaluation.

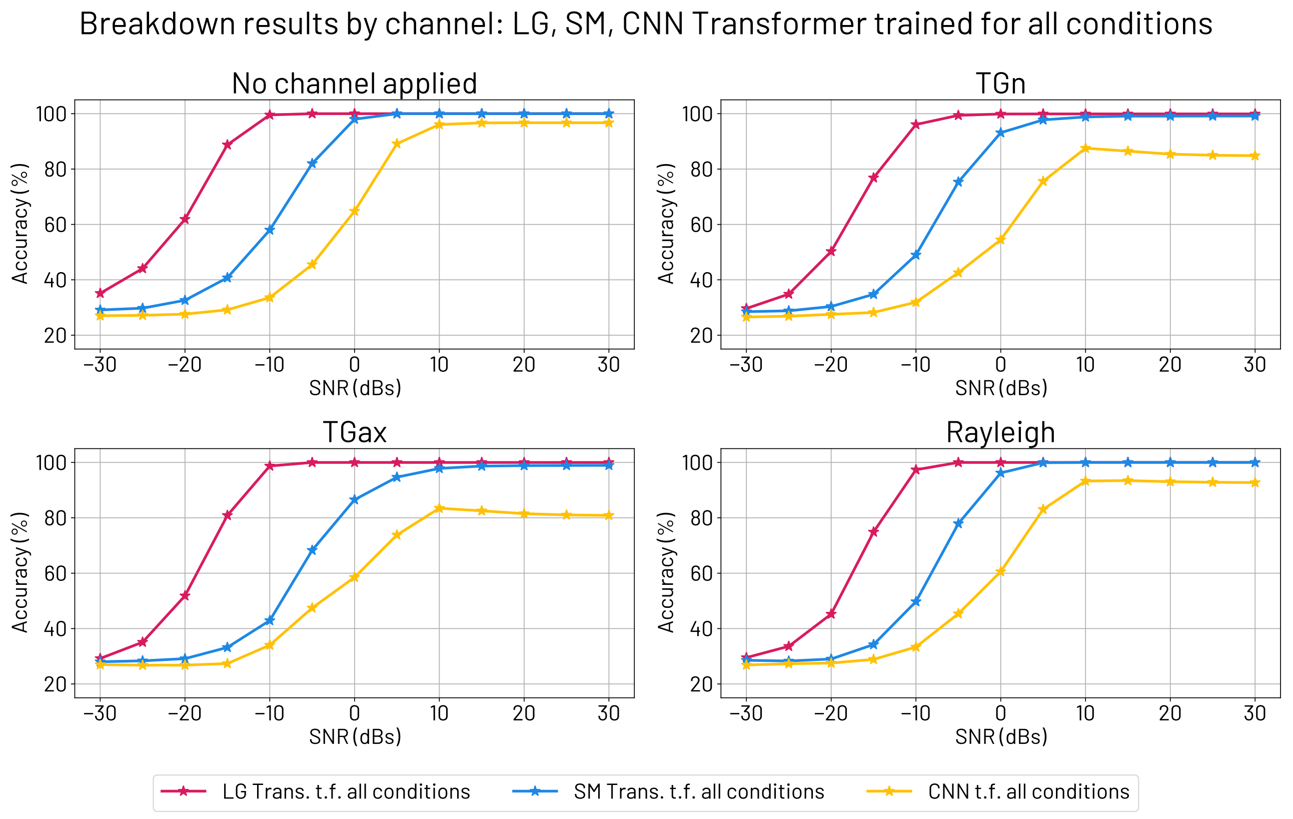


Figure 14: Channel performance breakdown for all architecture trained on all of the channels simultaneously and SNR ranging from -20 dBs to 30 dBs.

The final results of this experiment are presented and summarized in Figure 14. Upon analyzing the results, it becomes evident that the LG Transformer-based model exhibits superior performance compared to the other two models. This architecture demonstrates its ability to effectively learn and capture the intrinsic patterns in the data, regardless of the presence of noise or varying channel conditions.

The SM Transformer-based model secures the second position in terms of classification accuracy. It achieves near-perfect accuracy for all channel conditions; however, it demonstrates this remarkable performance primarily at higher SNR levels. As the noise level decreases, its accuracy consistently improves, reaching around 90% accuracy at 0 dB SNR and continuing to increase as the noise level further decreases.

On the other hand, the CNN-based model performs comparatively lower than the other two architectures. It fails to reach the same level of accuracy as the LG and SM Transformer models for all channel conditions, and specially for both TGn and TGax channel conditions. Moreover, the CNN-based model struggles the most when confronted with higher noise levels, resulting in reduced prediction accuracy.

These findings highlight the varying strengths and weaknesses of each model type in the context of our experiments. The LG Transformer model demonstrates robustness and adaptability across different channel conditions but only when presented with enough data, while the SM Transformer model excels in achieving high accuracy with lower noise levels and less data, but it is not so robust to noise. However, the CNN-based model lags behind in terms of performance, particularly when faced with challenging noise environments.

4.4 Global comparison

In this final subsection of the results, we will provide a horizontal overview of the obtained results across the different experiments. Specifically, we will focus on comparing the average accuracy between the two top-performing architectures, namely the LG and SM Transformer-based models. In the previous Figure we can see clearly that the CNN model does not keep up in terms of classification accuracy. Additionally, we will conduct a comprehensive comparison of all architectures using various metrics, including model size, inference time, and classification accuracy.

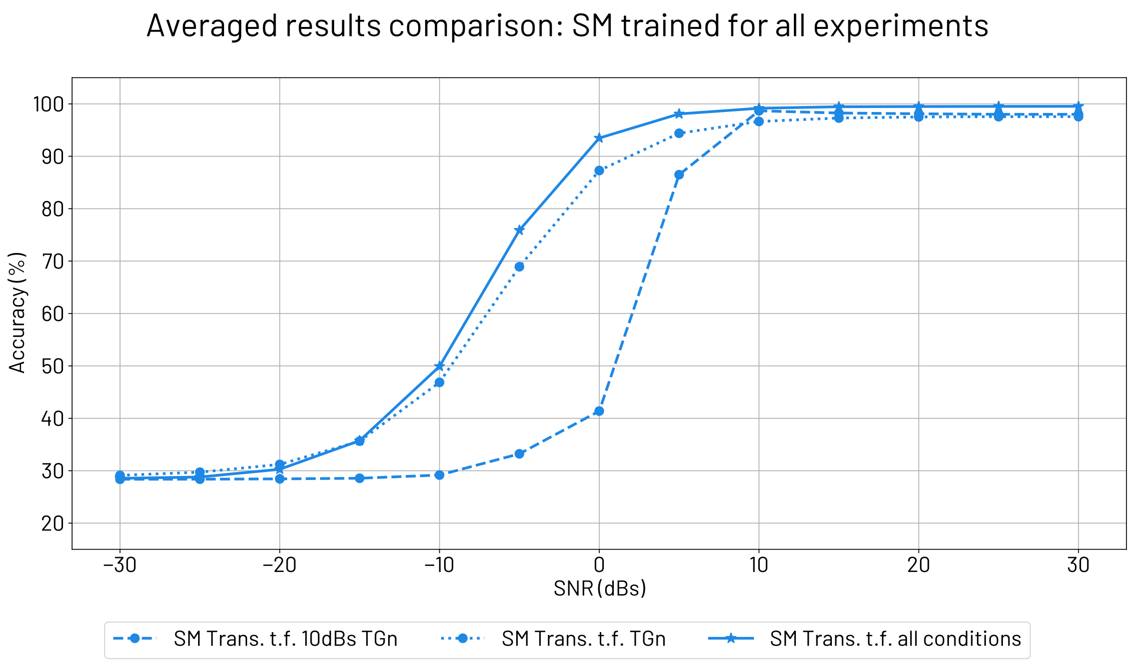


Figure 15: Comparison in average performance by a SM model belonging to each experiment. For the first two experiments, we have selected the model trained for TGn channel.

Figure 15 provides a comprehensive overview of the results obtained for the SM Transformer-based model across the three experiments. It allows us to examine the model's performance in terms of noise and channel conditions, providing valuable insights into its ability to generalize and adapt.

Looking at the graph, we can observe that the model trained in the third experiment (represented by the solid line) consistently outperforms the models from the first two experiments. This validates the hypothesis that exposing the model to a wider range of data, encompassing different noise levels and channel conditions, enhances its generalization capabilities. The improved performance of the third experiment model suggests that training with diverse data helps the model better handle the challenges posed by real-world scenarios.

While the third experiment model excels, it is interesting to note that the model from the second experiment (represented by the dotted line) is not far behind. This model, trained with variable noise levels but a specific channel condition, demonstrates good performance across a range of noise levels. Its ability to adapt to different noise levels during training contributes to its resilience in handling noise variations during testing. In contrast, the model from the first experiment (represented by the dashed line) exhibits the lowest performance, indicating that training with a fixed noise level and a single channel condition limits the model's ability to generalize effectively.

For the first two experiments we have picked the models trained for TGn channel condition. Throughout the experiments, models trained on the TGn channel environment consistently demonstrate better generalization compared to other channels. This finding, supported by experimental evaluations, highlights the significance of selecting appropriate channel conditions during training to improve the model's overall performance.

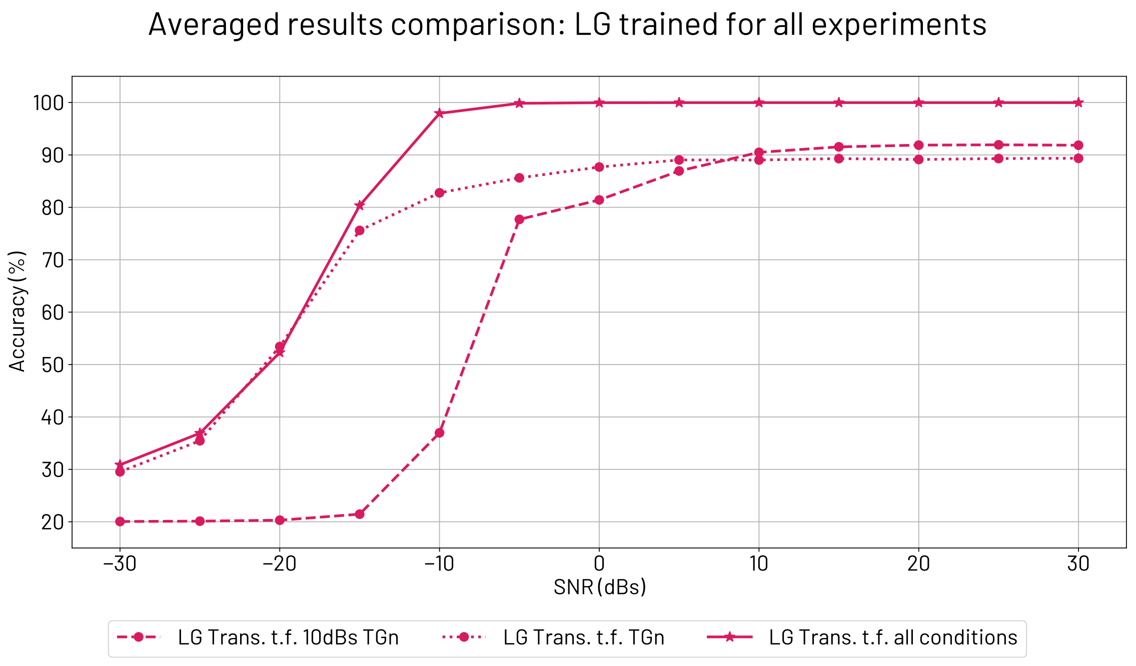


Figure 16: Comparison in average performance by a LG model trained in each of the different settings or experiments. For the first two experiments, we have selected the model trained for TGn channel.

When analyzing Figure 16, we can observe similar trends to the previous graph, but with some notable differences. In this case, the model trained in the third experiment (represented by the solid line) shows a more significant deviation from the models trained in the first two experiments. While the first two experiments yield models that reach a maximum accuracy of around 90%, the model trained with all conditions achieves nearly perfect classification accuracy. This suggests that augmenting the dataset with noise and different conditions has a significant impact on the model's ability to handle and classify signals in all environments.

Interestingly, the lines representing the models trained in the last two experiments, which were exposed to varying noise levels, exhibit closer proximity to each other for lower SNR values. This implies that training with varying noise levels helps the models better adapt to and classify signals in challenging noise conditions. Moreover, the model trained in the third experiment, which experienced all conditions during training, consistently outperforms the other models across all SNR levels.

It is important to note that the LG Transformer-based model does not achieve perfect classification accuracy in the first two experiments. This suggests that the model may be overfitting to the training data and failing to learn generalized features that can effectively identify different protocols under all conditions. This discrepancy highlights the importance of carefully selecting training strategies and ensuring models are capable of capturing and generalizing relevant signal patterns.

We have seen how the models trained for all conditions continue to outperform the models trained in

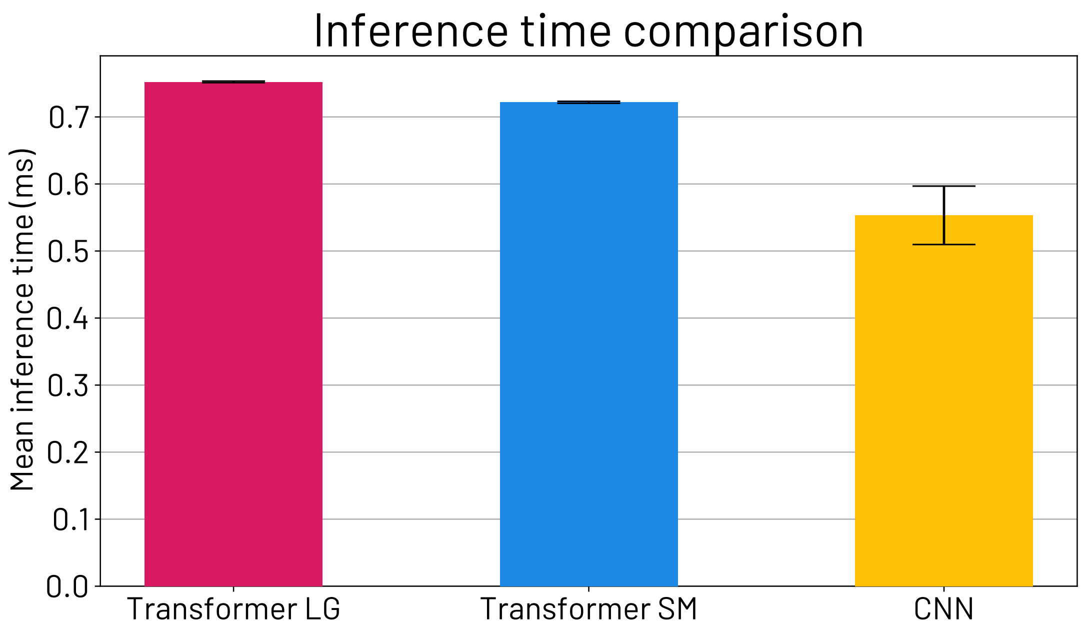


Figure 17: Comparison between models in terms of inference time. Time unit is milliseconds.

more specific environments, even in the conditions that those models were trained on. Now, if we look at Figure 17, we can observe the difference in inference time for each of the architectures trained. The inference times were measured using an `NVIDIA GPU`, starting with a warm-up phase and then performing 300 repetitions of the inference process to obtain precise timings.

According to the results, the CNN-based model demonstrates the fastest inference time, with a difference of more than 0.1 ms compared to the other architectures. This indicates that the CNN architecture is efficient in terms of computational speed, making it a favorable choice for applications that require real-time or low-latency processing.

On the other hand, both Transformer-based architectures, despite having a significant difference in the number of parameters, exhibit similar inference times. Both models have an average inference time slightly above 0.7 ms, with the smaller version of the Transformer showing slightly faster inference speed. This suggests that the Transformer architectures may introduce additional computational overhead due to their complex architecture and attention mechanisms.

In summary, Figure 17 highlights the trade-off between model architecture and inference time. The CNN-based model offers the fastest inference speed, making it suitable for applications where real-time processing is crucial. Meanwhile, the Transformer-based models, although slower in inference, provide the advantage of capturing long-range dependencies and context in the data, which can be beneficial for tasks that require a deeper understanding of the input or more precise output.

When examining Table 3, we can gain further insights into the performance of the different architectures across various metrics. The table provides accuracy measurements at three different SNR levels (-10 dBs, 0 dBs, and 10 dBs), as well as other metrics related to model size and speed.

In terms of classification accuracy, the LG Transformer-based architecture emerges as the clear winner, demonstrating superior performance across all SNR levels. The other Transformer-based architecture also

Architecture type	**Num. params**	**IQ samples / input**	**Inference time (ms)**	**Accuracy at -10 dBs**	**Accuracy at 0 dBs**	**Accuracy at 10 dBs**
CNN	4.1M	**512**	$\mathbf{0.5533 \pm 0.04357}$	33.22%	59.58%	90.09%
SM Transformer	**1.6M**	1536	0.7219 ± 0.00145	49.91%	93.48%	99.17%
LG Transformer	6.8M	8192	0.7523 ± 0.00116	**97.93%**	**99.97%**	**99.98%**

Table 3: Summary of models analyzed. We are considering the models trained for the last experiment, namely trained for multiple channel and noise conditions.

performs well in terms of accuracy, particularly at higher SNR levels. However, it is important to note that the CNN model sacrifices accuracy to some extent, as it achieves much lower accuracy values compared to the Transformer-based models.

Considering model size, the SM Transformer-based architecture stands out with the smallest number of parameters. It has approximately 2.5 million fewer parameters compared to the CNN model and 5.2 million fewer parameters compared to the LG Transformer model. This reduced parameter count can be advantageous when deploying the model on devices with limited memory or storage capacity, such as edge devices.

When evaluating inference speed, the CNN model proves to be the most efficient. It requires fewer IQ samples for inference, with a factor of 3 fewer samples compared to the SM Transformer model and 16 fewer samples compared to the LG Transformer model. Additionally, the CNN model demonstrates the quickest inference time, surpassing the other architectures by more than 0.1 ms.

5. Conclusions and Future Work

In terms of efficiency, the baseline CNN model is still a top contender, showcasing remarkable sample efficiency and inference speed. However, it is important to note that this efficiency comes at the expense of highly reduced accuracy compared to the Transformer-based architectures. As we were able to see in the third experiment, the generalization capacity of this type of NN is not at the same level as the Transformer-based models.

Ultimately, the selection of the most suitable architecture depends on the specific requirements of the application at hand and the available deployment resources. If accuracy is of paramount importance, the LG Transformer model is the most reliable choice. For scenarios where limited model size is crucial, the SM Transformer model provides a more space-efficient solution. On the other hand, if sample efficiency and rapid inference are top priorities, the CNN model offers a compelling option. It is essential to strike a balance between these factors to ensure optimal performance and resource utilization in real-world applications.

Indeed, our designed solution, whether it be the SM or LG Transformer-based models, has demonstrated superior performance compared to the baseline model that has been prevalent in wireless applications over the past years. These newer architectures have showcased their usefulness even in limited capacity scenarios, as exemplified by the SM model. Both models achieved remarkable accuracy, nearing perfection, when trained under various conditions and for high Signal-to-Noise Ratio (SNR). Additionally, they performed exceptionally well even in the presence of high noise, as evidenced by the LG model achieving a performance of nearly **97.93%** at -10 dB SNR.

We also conducted an analysis to examine the impact of three common channel models on base signals, highlighting the significant challenge they pose to waveform classification due to the complete alteration of wireless signal shape and dynamic range. Throughout the course of our experiments, we progressively augmented the training dataset with additional scenarios, allowing us to observe how the models gradually learned to extract intrinsic patterns specific to each protocol, while remaining invariant to channel and noise variations. This was particularly evident as the models encountered more diverse and challenging samples.

It is worth noting that our LG Transformer-based model successfully met all defined project specifications. It maintained an inference time under 1 ms, utilized 6.8 million parameters, achieved high classification accuracy at low SNR levels, and near-perfect accuracy at high SNR levels. Furthermore, it demonstrated consistent performance across different channel conditions, despite exclusively relying on synthetic data augmented during the training process. The SM version also fulfilled all specifications, with the exception of achieving high accuracy at low SNR levels, as its performance declined more rapidly with increasing noise levels.

In summary, we have been able to see a comprehensive comparison of the different architectures based on accuracy, model size, and speed. The LG Transformer model excels in terms of accuracy, while the SM Transformer model offers a more compact model size. The CNN model showcases the best properties in terms of sample efficiency and inference speed, although with a trade-off of reduced accuracy compared to the Transformer-based architectures. The choice of architecture depends on the specific requirements of the application and the available resources for deployment. However, it is fair to claim that the attention mechanism employed in Transformer-based models plays a pivotal role in significantly improving the performance of our classifiers. This mechanism allows the models to effectively model relationships and dependencies between different elements of our input sequences. As a result, Transformer architectures have demonstrated superior performance, surpassing very established architectures such as CNNs.

5.1 Future Work

The results and models presented in this thesis were obtained using synthetic data generated through `MATLAB` and augmented in `Python`. The next crucial step is to evaluate the performance of the trained classifiers using real over-the-air (OTA) data. This evaluation will provide valuable insights into how well the models generalize to real-world scenarios. Fine-tuning approaches can be employed to further enhance performance in specific scenarios, reducing the need for extensive collection of OTA data, which can be expensive and time-consuming compared to simulating Wi-Fi transmissions.

Another interesting next step, would be to enable real-time inference. To do so, a comprehensive pipeline needs to be implemented. This involves collecting IQ samples using a receiver antenna, recording an adequate amount of data for the model to perform inference, and obtaining predictions for the transmitted signal. This real-time inference capability would be essential for practical deployment of the waveform classification system. Also, a graphical interface to visualize the predictions could be implemented. The way in this could be done is included into the Appendix A.

Another compelling research direction is to assess how the trained classifiers perform in the presence of overlapping protocols. This can be accomplished by utilizing one of the pretrained classifiers with a modified output layer that employs a Sigmoid activation function per output neuron instead of a LogSoftmax. This modification allows for independent probabilistic predictions for each protocol, enabling the identification of multiple simultaneous transmissions.

In addition, given the significant impact of attention mechanisms and Transformer models on the classifiers performance, it would be intriguing to visualize the effect of the Transformer model by examining the interactions between keys and queries in the attention operation. This visualization could provide valuable insights into which parts of the waveform are given more importance and play a crucial role in selecting the transmitted protocol. This interpretability can enhance the security and trustworthiness of the machine learning approach, not only in wireless communications but also in various other domains where this property is desired.

This project, focused on waveform classification using deep neural networks, serves as an exemplar of how machine learning modules can be seamlessly integrated into wireless communications applications to provide real-time or near-real-time services, such as protocol identification. It showcases the potential of machine learning techniques to enhance and extend the capabilities of traditional wireless communication systems, opening doors to a wide range of innovative applications in the field.

Biblography

References

[1] Wikipedia: Rayleigh fading, Aug 2022.

[2] Ahmed Alkhateeb, Sam Alex, Paul Varkey, Ying Li, Qi Qu, and Djordje Tujkovic. Deep learning coordinated beamforming for highly-mobile millimeter wave systems. *IEEE Access*, 6:37328–37348, 2018.

[3] Daoud Burghal, Ashwin T. Ravi, Varun Rao, Abdullah A. Alghafis, and Andreas F. Molisch. A comprehensive survey of machine learning based localization with wireless signals, 2020.

[4] Jingjing Cai, Fengming Gan, Xianghai Cao, and Wei Liu. Signal modulation classification based on the transformer network. *IEEE Transactions on Cognitive Communications and Networking*, 8(3):1348–1357, 2022.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[6] Vinko Erceg. IEEE P802.11 Wireless LANs TGn Channel Models. 2004.

[7] Janne M. J. Huttunen, Dani Korpi, and Mikko Honkala. DeepTx: Deep learning beamforming with channel prediction, 2022.

[8] J.P. Kermoal, L. Schumacher, K.I. Pedersen, P.E. Mogensen, and F. Frederiksen. A stochastic MIMO radio channel model with experimental validation. *IEEE Journal on Selected Areas in Communications*, 20(6):1211–1226, 2002.

[9] Fan Meng, Peng Chen, Lenan Wu, and Xianbin Wang. Automatic modulation classification: A deep learning enabled approach. *IEEE Transactions on Vehicular Technology*, 67(11):10760–10772, 2018.

[10] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, and Eduardo Pasiliao. Detection of rogue RF transmitters using generative adversarial nets. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7, 2019.

[11] Kunal Sankhe, Mauro Belgiovine, Fan Zhou, Shamnaz Riyaz, Stratis Ioannidis, and Kaushik Chowdhury. ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks, 2018.

[12] Nasim Soltani, Debashri Roy, and Kaushik Chowdhury. PRONTO: Preamble overhead reduction with neural networks for coarse synchronization. *IEEE Transactions on Wireless Communications*, pages 1–1, 2023.

[13] Jiyun Tao, Jienan Chen, Jing Xing, Shengli Fu, and Junfei Xie. Autoencoder Neural Network Based Intelligent Hybrid Beamforming Design for mmWave Massive MIMO Systems. *IEEE Transactions on Cognitive Communications and Networking*, 6(3):1019–1030, 2020.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[15] Qingyang Wu, Carlos Feres, Daniel Kuzmenko, Zhi Ding, Zhou Yu, Xin Liu, and Xiaoguang Liu. Deep learning based RF fingerprinting for device identification and wireless security. *Electronics Letters*, 54, 12 2018.

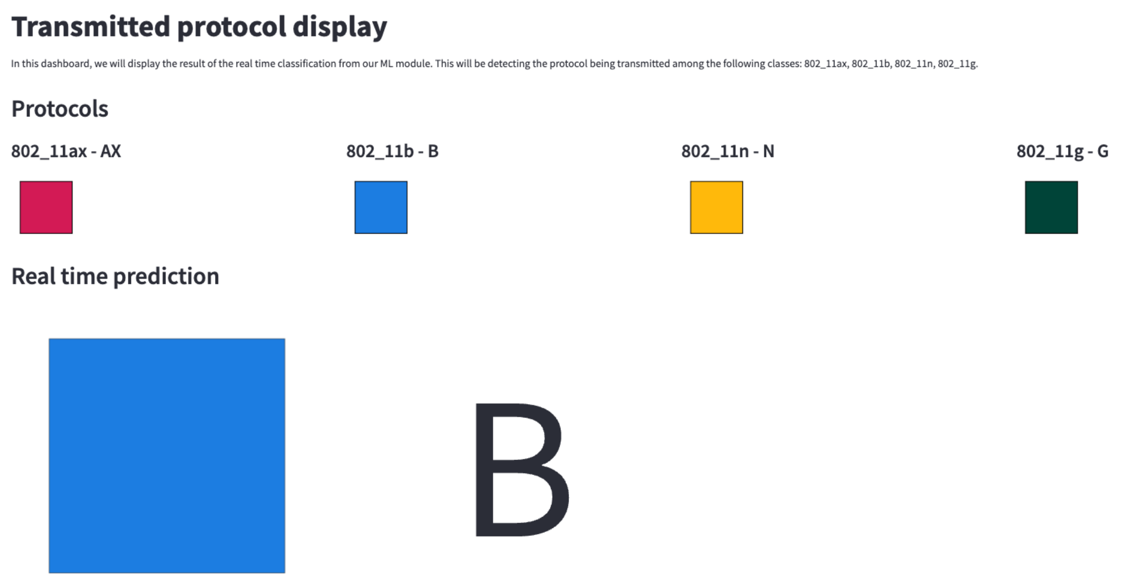
[16] Ke Yang, Sixian Wang, Jincheng Dai, Kailin Tan, Kai Niu, and Ping Zhang. WITT: A Wireless Image Transmission Transformer for Semantic Communications. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

Appendix

A. Streamlit Demo

To facilitate real-time inference of a trained and deployed model, we developed a user-friendly Graphical User Interface (GUI). This intuitive display offers a dual functionality, providing immediate visibility of the current identified protocol while also presenting a historical perspective with the last 300 predictions. By offering this comprehensive view of the model's performance over time, users can effectively evaluate classifier performance, track trends, and gain valuable insights into the classification process. The GUI serves as a powerful tool for assessing and monitoring the model's classification capabilities in real-world scenarios.

As previously described, the GUI comprises two distinct components. The first component prominently displays the latest model prediction, utilizing a color box and corresponding letter to indicate the identified protocol for the input IQ samples. This functionality, illustrated in Figure 18, is accompanied by a concise explanation and a legend that establishes the color-protocol mapping. The enlarged size of the color box and letter for the current prediction serves to capture the user's attention, emphasizing the real-time aspect of the prediction and enhancing its visibility for improved user experience.


Transmitted protocol display

In this dashboard, we will display the result of the real time classification from our ML module. This will be detecting the protocol being transmitted among the following classes: 802_11ax, 802_11b, 802_11n, 802_11g.

Protocols

802_11ax - AX	802_11b - B	802_11n - N	802_11g - G
🟥	🟦	🟨	🟩

Real time prediction

🟦 B

Figure 18: The first part of the inference Graphical User Interface (GUI) includes a concise dashboard explanation, a legend associating protocols with colors, and a prominent display of the latest model prediction for real-time feedback.

The second component of the GUI is the historical view of the last 300 predictions, represented as a timeline. Each slice on the vertical axis represents a protocol, indicated by its specific color. The horizontal axis represents time, with older predictions on the left and newer predictions on the right. As the model continues to make predictions, the timeline moves to the left, removing the oldest predicted protocols. Figure 19 provides an example of this visualization, demonstrating how the timeline dynamically updates as new predictions are made.

Furthermore, it is noteworthy to mention that the chosen color palette in the GUI is designed to be colorblind friendly, ensuring accessibility for individuals with color vision impairments. By adopting a

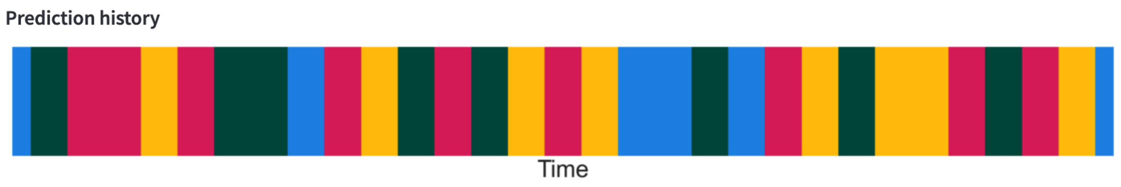


Figure 19: The second part of the GUI features a prediction history visualization, consisting of a colored line graph that represents the model's last three hundred predictions. This visual representation allows users to track the patterns and evolution of the model's predictions over time.

color scheme that accommodates diverse visual abilities, the visualization of colors remains inclusive and understandable to all public.