POLITECNICO DI TORINO

Dipartimento di Elettronica e Telecomunicazioni

Master's Degree in
Communications and Computer Networks Engineering

Master's Thesis

Deep Learning Application to 5G Physical Layer for Channel Estimation and CSI Feedback Improvement

Supervisors
Prof. Roberto Garello
Prof. Monica Visintin
Eng. Roberto Fantini

Candidate
Elisa ZIMAGLIA

ACADEMIC YEAR 2018 - 2019

To my grandparents

Acknowledgements

I would like to spend some words to thank all those people who contributed in different ways to the achievement of this important success.
The biggest "thanks" goes to my family. By offering me their constant support and confidence, my parents have fueled my ambitions and helped me gaining the master degree title. Thanks also to my sister Arianna, who has always been my point of reference. Everytime I need, I know I can rely on her completely.
I thank all my classmates, with whom I have not only shared exams, lessons and school projects, but also a lot of amusing and relax moments that made my years as a Politecnico student a little bit lighter.
I also owe a lot to all my dear friends from Alba and Asti, because they have always been able to fill my little spare time with laughter and good memories.
Another thanks is dedicated to my colleagues, for never denying me their help when I needed it, but most of all for helping me keeping things light.
Thanks to Andrea, who, from being my deskmate on the first day of lesson, has finally turned into a special friend and a precious partner in study and laughs.
Another sincere "thanks" goes to my old friends Giulia and Maura, who have been next to me through all the wins and difficulties. I knew I could always count on them for a good advice, for some comfort, or even better for some laughs together.
I certainly owe a special thought to Lorenzo, who has always been by my side during the last years, transmitting me the confidence and strength that I'm not always able to find by myself.
Of course, I cannot forget to thank Ilaria who, from a simple roommate, has gradually turned into a life partner, my anchor, without whom these years would have not been the same.
I thank my supervisor, Professor Garello, for his willingness to help me giving form to the present thesis project. Together with him, I also owe a dutiful "thanks" to Professor Visintin and Giulio, who have somehow contributed to my researches and to my work.
Last but not least, I would like to thank my tutor, Roberto Fantini, whom I have learned to value both as a professional and as a person. His extreme availability and his precious guide and advice have been fundamental for me to carry out this project.

Abstract

Starting from the last decades of the 20th century, Machine Learning has been widely applied in many engineering fields, such as communications, speech and image processing, computer vision and robotics, resulting particularly effective and useful in contexts where a rigorous mathematical model of the problem is too hard to be elaborated. Focusing on wireless communication systems, in recent years Machine Learning applications to the upper layers have been minutely explored for various purposes, like the deployment of cognitive radio and Self Organized Networks or the resource management, while its application to the physical layer has been somehow overlooked.

The purpose of this thesis is to investigate the potential use of neural networks for the optimization of specific physical layer blocks in a communication system, taking into account the peculiar characteristics of the emerging radio technologies based on 5G standard (e.g. massive Multiple-Input Multiple-Output, beamforming, millimeter Waves) and all their related challenges. In particular, the analysis focuses on channel estimation and Channel State Information feedback reporting blocks, providing data and statistics that are representative of how Machine Learning algorithms introduction affects radio link performance, in terms of BLock Error Rate and throughput.

Contents

List of Tables

List of Figures

Chapter 1

Machine Learning: an overview

This chapter gives an overview of Machine Learning, distinguishing among different approaches and providing definitions and examples. First of all, it is important to highlight that Machine Learning is a subfield of Artificial Intelligence: reference [14] explains that, while the term **Artificial Intelligence (AI)** gathers all the forms of technologies that hold some intelligence, Machine Learning identifies a specific group among these technologies.
There are plenty of definitions for Machine Learning in literature: the authors of book [16] describe it as a set of techniques designed to automatically recognize patterns in data and then use those patterns to predict other data or to make some kinds of decision. The key idea behind Machine Learning is that the algorithm has to figure out the model autonomously, starting from some data (e.g. images, audio, documents) [16]. In fact the name "Machine learning" suggests that there is no human component involved: the technique itself analyzes the set of available data, called **training data**, and "learns" how to elaborate the correct **model** when laws and logical reasoning are not promising.

1.1 Types of machine learning

Machine Learning (ML) techniques can be classified as **supervised**, **unsupervised** or **reinforcement** learning. These three learning types present different characteristics and each of them has its usual domain of application, even if they are not completely formal and distinct concepts.

1.1.1 Supervised or predictive learning

In supervised approach, each training dataset consists of an input and the corresponding correct output, which is what the learned model is supposed to produce

$$\{input, correct\ output\}.$$

This is the most widely used form of machine learning, because it is the most similar to the process by which humans learn things. To clarify this idea, reference [14] provides an effective analogy: when humans want to find the solution to an exercise, they apply their current knowledge to solve the problem and compare the results with the correct solutions: if the answer is wrong, they take note of this, modifying their knowledge about the problem and then repeating previous steps.
The fundamental property of this learning approach is the knowledge of desired outputs: in fact the name itself suggests a form of tutoring, like when the teacher provides students with solutions that they need to memorize [14]. In principle, the output of the model can be anything, but in most cases it is:

- a **categorical** or **nominal** variable from a finite set; in this case the problem is referred to as **classification** or **pattern recognition**;
- a real scalar value; in this case the problem is referred to as **regression**.

Classification

The goal of classification is to learn how to map the input x into the output y, where $y \in \{1, ..., C\}$ and C is the number of possible classes which the input can be assigned to. When $C = 2$, the problem is referred to as **binary classification**; if instead $C > 2$ it is called **multiclass classification** [16].
These class **labels** are discrete and unordered values [19] and the model aims at assigning them to new input instances, based on past observations.
Reference [14] provides some example of classification problems:

- spam mail filtering: the model classifies the mails as regular or spam;

- digit recognitions: the model classifies the digit image into one number ranging between 0 and 9;
- face recognition: the model classifies the face image selecting the right registered user.

Regression

The main difference with respect to classification is that the regression outcome is continuous. As book [19] explains, regression analysis is characterized by predictor or **explanatory** variables and a continuous response variable (**target** or **outcome**): the model aims to find a relation between those two sets of variables and to predict an output.
In reference [16], several examples of real-world regression problems are provided:

- to predict the age of a viewer watching a given video on YouTube;
- to predict the temperature at some location within a certain area exploiting weather data, time, sensors etc.
- to predict tomorrow's stock price given current market conditions and other variegated information.

1.1.2 Reinforcement learning

The goal of this machine learning approach is to develop a system, called **agent**, able to improve its performance based on interactions with the **environment**. Actually, reinforcement learning is sometimes labeled as a subfield of supervised learning, since it includes a **reward signal** [19]. However, book [16] clarifies that there exists a fundamental difference between this kind of technique and the canonical supervised approach described in previous sections: reward signal is not the correct value or label, it is simply a measure that tells how good or bad an action is considered according to a reward function.
So training data contains only input, some output and the corresponding grade:

$$\{input,\ some\ output,\ grade\ for\ this\ output\}$$

Interacting with the environment, the agent gets back a **state** value and exploits reinforcement learning to select a series of actions that maximizes the reward [19]. This mechanism is effectively illustrated in the scheme in `Figure 1.1`.

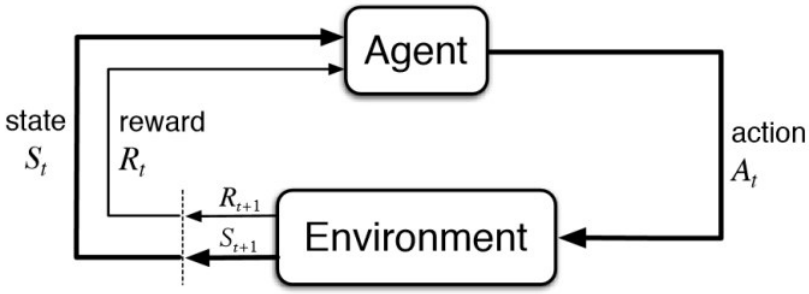


Figure 1.1: Reinforcement learning process
(taken from [W20])

Book [19] proposes a popular example of reinforcement learning application, that is chess engine: here, the agent decides a sequence of moves based on the state of the board and the reward can be identify in the win or defeat at the end of the game.

1.1.3 Unsupervised learning

This third approach has the goal of discovering interesting structures or patterns characterizing the input data and of preprocessing them [19]. The substantial novelty with respect to supervised learning approach is that it deals with unlabeled data or data with unknown structure in the form

$$\{input\}$$

Reference [19] explains that unsupervised learning must be able to explore the structure of our data to extract meaningful information without the guidance of a known outcome or reward function. In general, this makes the unsupervised problem significantly more complex with respect to the supervised learning case; however, unsupervised learning is the most widely applicable approach since it does not require data labeling process, that is often expensive and inefficient.

Clustering

The problem of clustering data into groups is a typical example of unsupervised learning. Information is organized into meaningful subgroups, called **clusters**, without any prior knowledge about their group memberships; each cluster encloses elements that share a certain degree of similarity that is higher then the one shared with members of other groups [19]. `Figure 1.2` shows how data can be clustered into two different groups based on the similarity of their two features x and y.

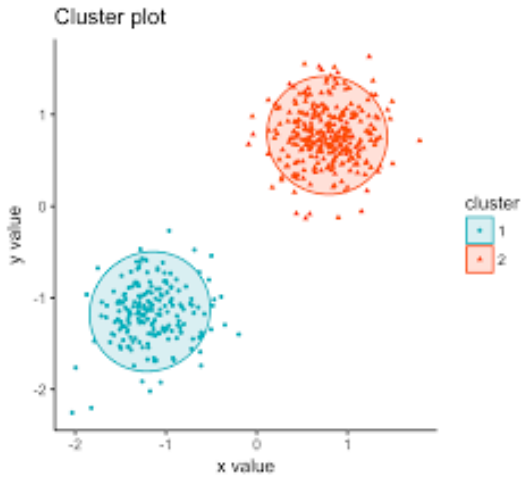


Figure 1.2: Example of clustering
(taken from [W26])

Some real-world applications of clustering are illustrated in reference [16]:

- in e-commerce, a worldwide used approach reckon on clustering the users into groups, based on their web-surfing trend and their purchasing habits, aiming at wisely targeting different advertisements;

- in biology, it can be useful to group flow-citometry data, to discover different sub-populations of cells.

Data compression through dimensionality reduction

This kind of unsupervised learning is based on the concept of **dimensionality reduction**: projecting data to a lower dimensional subspace, sufficient to capture their essential features, can result in a significant advantage from a complexity point of view, like authors of [16] explains; however, complexity reduction is not the only advantage obtained. In fact, according to reference [19], this kind of technique is commonly used in feature preprocessing, to remove the noise which degrades predictive performance of certain ML algorithms.
Book [16] justifies this by clarifying the concept of **latent factor**: even if a set of data can appear high-dimensional, the degrees of variability (called latent factors) could result being very few. Low dimensional representations often lead to better accuracy in predicting tasks, since they focus on the "essence" of the object and ignore misleading information which acts as noise. Clearly a good trade off between simplicity and meaningfulness must be found, in order to obtain an effective model.

Principal Component Analysis (PCA) is one of the possible approaches to dimensionality reduction. Book [16] provides some examples of PCA applications:

- in signal processing, a variant of PCA is used to separate signal into their different sources;
- in computer graphics, motion capture data can be reduced to low dimensional representation, with the purpose of creating animations.

1.2 Basic concepts about ML

Previous sections have simply provided a general description of how ML works and a list of the possible learning approaches with their relative use cases. Anyway, it is important to understand which are the fundamental challenges of ML algorithms that sometimes may lead to prefer a classical mathematical approach.

1.2.1 Generalization

Generalization is a key concept, defined in reference [14] as the process used to make the model performance independent from the training data and from the input.
This independence is important if one thinks to the machine learning process, illustrated in `Figure 1.3`: it is clear from the scheme that data used to train the model and data that feed the field application are distinct.

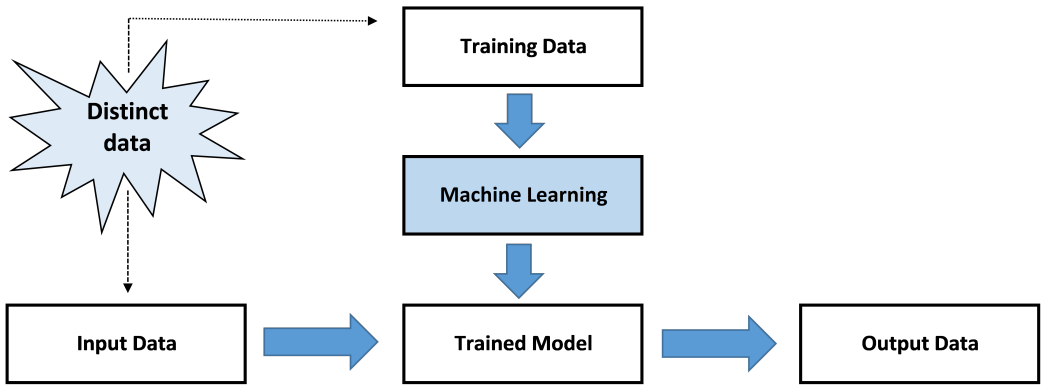


Figure 1.3: Applying a ML model
(taken from [14])

The distinctness between training data and input data is a fundamental challenge that every ML algorithm has to deal with: it is very unlikely that a model trained

by handwritten digits from a single person will be able to recognize digits written by other people, since the algorithm will probably learn some writing stokes that are peculiar of the single person calligraphy but completely useless and even misleading for the recognition process [14]. This considered, training the model with proper data, sufficiently varied and representative of most input characteristics, is an essential starting point that impacts heavily on performance.

1.2.2 Overfitting

Overfitting is the main cause of contamination of generalization process. To clarify this concept, it is useful to consider a classification problem: a ML algorithm has to divide the position data into two group (cross or circle), so the purpose is to identify a curve which marks out the borders of the two groups, working on the training data.

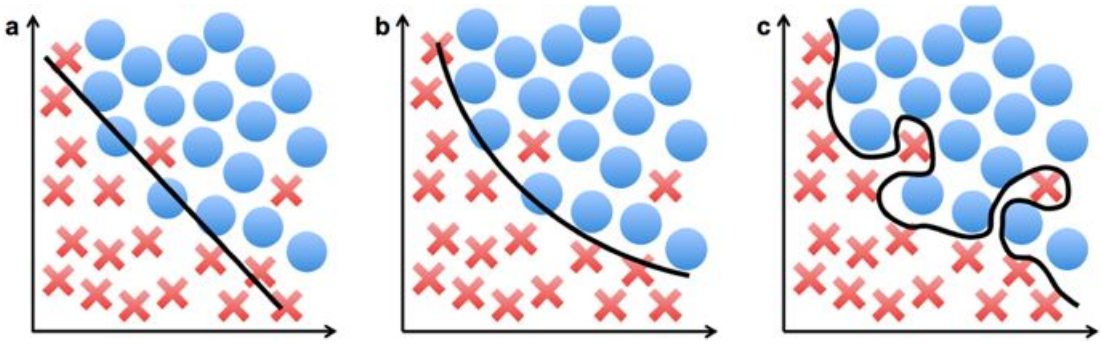


Figure 1.4: Example of overfitting
(taken from [W21])

Looking at `Figure 1.4`, it can be noticed that the first plot on the left, where the curve is a simple linear function, presents a significant number of outliers; this number decreases in the second plot, where a more complex curve is adopted as group delimiter. The last plot on the right instead shows a very complex curve that perfectly groups the points and thus yields optimal performance for the training data. The problem arises when the model is applied to input data different from the ones used for training: in that case, a new position can be classified in the wrong group due to some outliers in training data, which have penetrated the area of the other group corrupting the boundary.
Summarizing, when training data are not perfect and contain much noise, the ML algorithm elaborates a model with low generalizability, since it considers all data in the same way, not distinguishing relevant information from noise. This dynamic is called overfitting.

According to [14], overfitting can have a considerable impact on ML performance; as

so, several techniques have been developed to face this issue:

- **regularization**: a numerical method that simplifies the model in order to prevent overfitting;
- **validation**: a process that splits the dataset in training and validation set: the first one is used to elaborate the model while the second one has the function of monitoring the performance.

The following section focuses on validation solution.

Validation

The idea behind validation is to devote a portion of training dataset to detect overfitting effects: a trained model can be considered overfitted if it turns out poor performance when applied to validation data. Reference [14] illustrates the steps of validation process:

- step 1: split data into training and validation set;
- step 2: train the model using training data;
- step 3: evaluate performance of the model on validation data: if the model works well, continue the training, otherwise modify the model and repeat from step 2.

In `Figure 1.5` a simple scheme shows the split-process of a dataset when validation is performed.

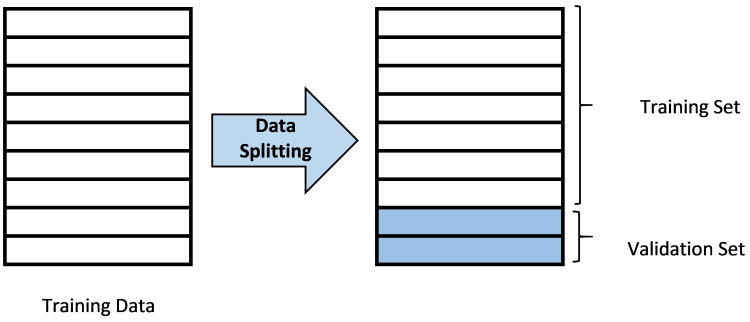


Figure 1.5: Splitting the dataset for validation
(taken from [14])

`Figure 1.6` depicts a splitting process that is slightly different: instead of fixing a split into training and validation data, the process continuously changes the dataset

division randomly. This procedure takes the name of **cross-validation** and results in a better ability to spot overfitting, especially when data at disposal are not so many and generalization process would be more challenging. In fact, when a portion of the dataset is affected by noise, even if the model trained over these corrupted data learns information that are meaningless and not generalizable, other portions of data are considered in the training process, so that the noise effect is attenuated.

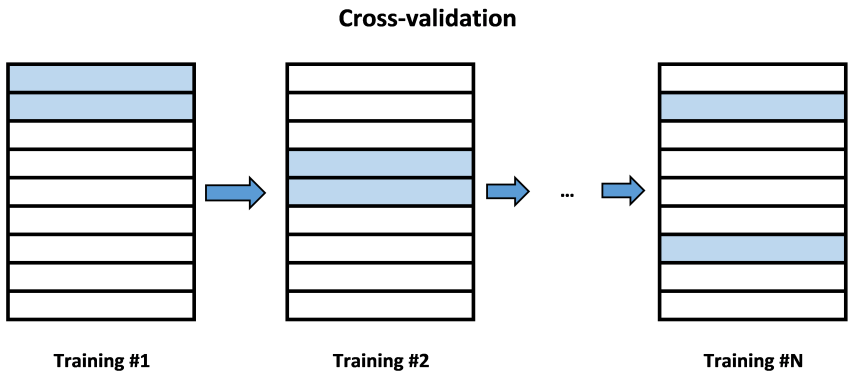


Figure 1.6: Splitting the dataset for cross-validation
(taken from [14])

1.3 Deep learning

Deep Learning (DL) is a kind of machine learning based on deep neural networks. **Neural Network (NN)** are nothing else than a possible implementation of ML models, where the process of determining the model takes the name of **learning rule**. The name suggests a reference to the mechanism of human brain: while the neurons transmit signals one to the other and their association gives form to information, in neural networks the **nodes** play the role of neurons, mimicing neuron's association by means of connection **weight** values [14]. The information of a neural network is collected inside weights and biases values, that are what ML algorithm needs to learn.

1.3.1 Neural network nodes

Figure 1.7 shows a node that receives three input signals x_1, x_2 and x_3; w_1, w_2 and w_3 are the **weights** associated to those inputs, while b is the **bias**.
The output y is computed starting from the weighted sum of the inputs and the bias

$$v = (w_1 \cdot x_1) + (w_2 \cdot x_2) + (w_3 \cdot x_3) + b \tag{1.1}$$

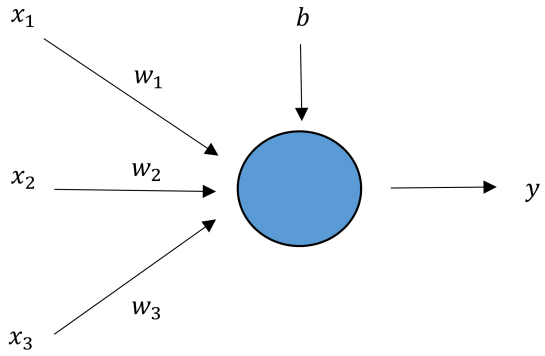


Figure 1.7: Node of a neural network
(taken from [14])

From (1.1) it is clear that each weight value determines how much impact the corresponding input has on the output: when a weight is null, the relative input is totally disconnected from the node. The weighted sum v is then passed to the **activation function** $\phi(.)$, in order to determine the output value

$$y = \phi(v) \tag{1.2}$$

1.3.2 Layered neural networks

Layered neural networks are the most widely used. As the name suggests, in this kind of networks the nodes are grouped in **layers**. Figure 1.8 shows an example of layered structure.

The group of yellow circles is called **input layer**, whose task is to transfer the input signals to the next nodes. Blue and green circles instead form the so called **hidden layers**, that precede the **output layer** in red. The signal enters the input layer, is transferred through the hidden layers and then reaches the output layer, that generates the outcome of the model.
Based on the number of hidden layers, it is possible to classify a neural network as:

- **single-layer neural network**, when input and output layers are directly connected;
- **shallow neural network**, if a single hidden layer is present;
- **Deep Neural Network (DNN)**, when the network contains two or more hidden layers.

Deep neural networks in turn can be distinguished in:

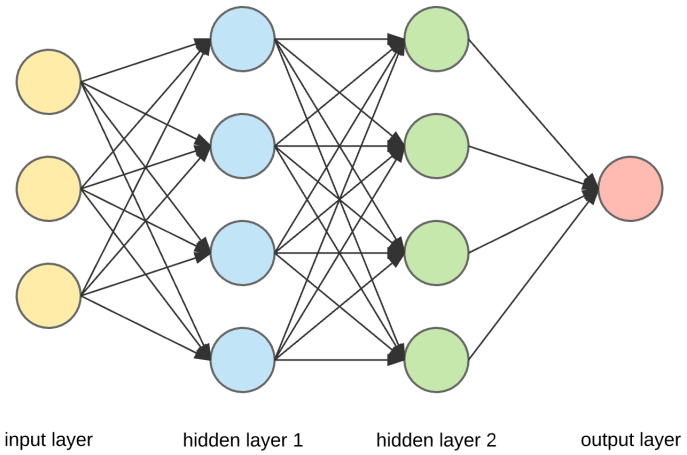


Figure 1.8: Layered neural network
(taken from [W15])

- **deep feedforward neural networks**: the term "feedforward" suggests the fact that there are no backwards connections by which outputs of the model are fed back to compute themselves: the information flows from the input to the output through the hidden layers, without inverting the propagation direction; the adjective "deep" instead stands for the considerable number of hidden layers that generally compose this kind of networks;

- **deep recurrent neural networks**: these networks are feedforward neural networks extended to include also feedback connections, so that loops are allowed;

- **deep convolutional neural networks**: these models are inspired by human brain mechanism for object recognition and are specialized for processing data characterized by a known grid-like topology, like images.

Convolutional neural network

In reference [12], a Convolutional Neural Network (CNN) is defined as a neural network that uses **convolution** in place of general matrix multiplication in at least one of its layers.
Deep convolutional networks build a **feature hierarchy** by combining in a layer-wise fashion the low-level features (i.e. extracted by early layers) to form high-level features [19]. Book [19] clarifies these concepts with some examples: low-level features can be identified in edges and blobs; their combination gives form to high-level features, like characterizing object shapes (building, car, tree, cat ...).

CNN effectiveness in image classification field is related to three key properties,

described in reference [W8]:

- **sparse connectivity**: instead of connecting every input to every hidden neuron, connections are limited to small regions (typically 3×3 or 5×5), called **local receptive fields**; this choice makes sense reasoning on the fact that in images content is usually "local", that means little correlation among pixels that are very distant. This local receptive field slides on the whole image and every position corresponds to a hidden neuron;
- **parameter-sharing**: all the connections from the local receptive field to each hidden neuron share the same weights and biases, which define a **filter**; as a consequence, all neurons in the same layer extract exactly the same feature at different locations in the image, so that a convolutional layer must consist of different filters, capturing distinct features. Feature extraction is performed through convolution operations between filters and local receptive fields, producing images that accentuates specific characteristics;
- **pooling**: pooling layers aim to simplify the output of a convolutional layer by performing a sort of downsampling; they are applied independently to each feature map. `Figure 1.9` reports an example of max-pooling layer, where each pooling unit outputs the maximum activation value in a 2×2 input region.

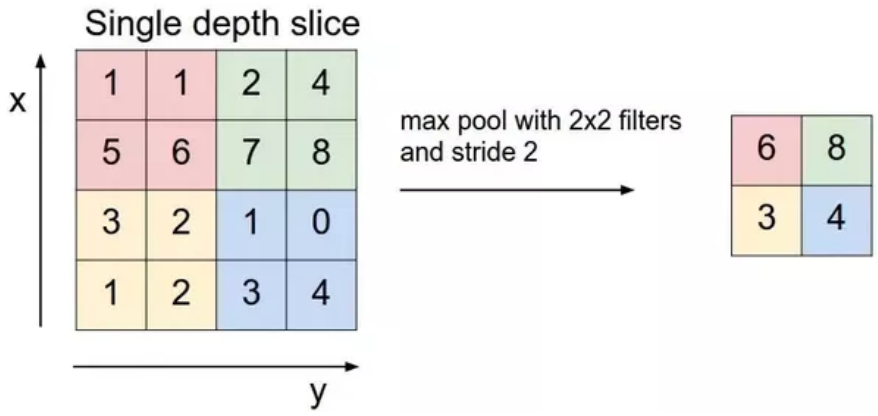


Figure 1.9: Example of max-pooling layer
(taken from [W12])

As reference [19] explains, these three fundamental features have an important consequence: the number of parameters learned by the network decreases significantly, that implies an improved ability in capturing essential features and a greater robustness against overfitting. In `Figure 1.10`, it is possible to observe the typical scheme of a convolutional neural network for image classification.

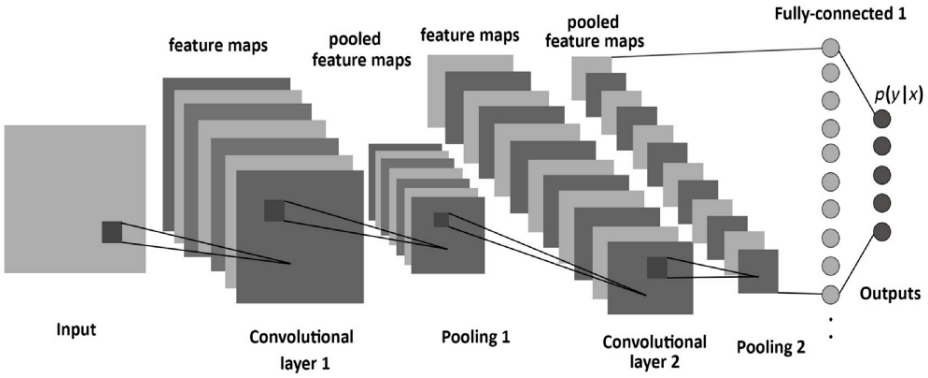


Figure 1.10: Example of CNN for image classification (taken from [W16])

In this structure, two different functional blocks can be distinguished:

- a **feature extractor**, composed of all the convolutional and pooling layers;
- a **classifier**, consisting in one or more fully connected layers.

Chapter 2

Introduction to 5G New Radio

In order to understand why 5G can be considered a turning point at the end of forty years marked by an inexorable technological evolution , it can be useful to outline the mobile communication context in which this new technology makes its debut.

The first generation made is entrance in the scenario of mobile communications around 1980. Working on analog transmission, the communication systems based on **1G** technology were limited to voice services and the service quality was quite low but, for the first time, mobile telephony was made accessible to ordinary people.
2G, appearing in the early 1990s, marked the introduction of digital transmission on the radio link. Voice traffic was still the target service, but the drop-out of analog transmission also enabled limited data services and significant improvements in terms of efficiency and security.The **3G** of mobile communication was introduced in the early 2000s. With 3G, wireless internet access became fast, high data rates were reached and multimedia services were made available. Starting from the years around 2010, **4G** became widespread, providing very high end-user data rates and Quality of Service (QoS) guarantees, entirely on IP traffic. Discussions on the **5G** of mobile communication systems began around 2012 and they are now finally turning into concrete realizations.

The second chapter has the purpose of providing an overview on 5G technology. The first section outlines the key novelties introduced by this new standard, listing the most evident points of divergence from 4G and the main use cases; in the second section instead the focus is on 5G physical layer: the idea is to give out a clear theoretical basis that is essential to understand the application of Machine Learning to New Radio physical layer, discussed in the third chapter.

2.1 From LTE to New Radio

The first release of Long Term Evolution (LTE) (Release 8) was in 2009 and from then on 4G standard has gone through several steps of evolution, bringing enhanced performance and extended capabilities. This includes, among all, features for enhanced mobile broadband (e.g. higher data rates, spectrum efficiency, coverage improvements) and important extensions of the set of use cases for which LTE was initially imagined [5]. With this in mind, the authors of reference [5] clarify that ongoing and future evolution steps of LTE should be considered an important part of the overall 5G radio-access solution, since 5G can be thought as a series of use cases to be supported rather than a specific access technology.

2.1.1 LTE advanced

After the starting releases 8 and 9 new requirements and expectations raised, thus following releases, known as **LTE advanced** or **evolution**, were prepared with the purpose of providing additional enhancements and features in different fields. Figure 2.1 shows some of the main areas in which LTE has evolved over several years.

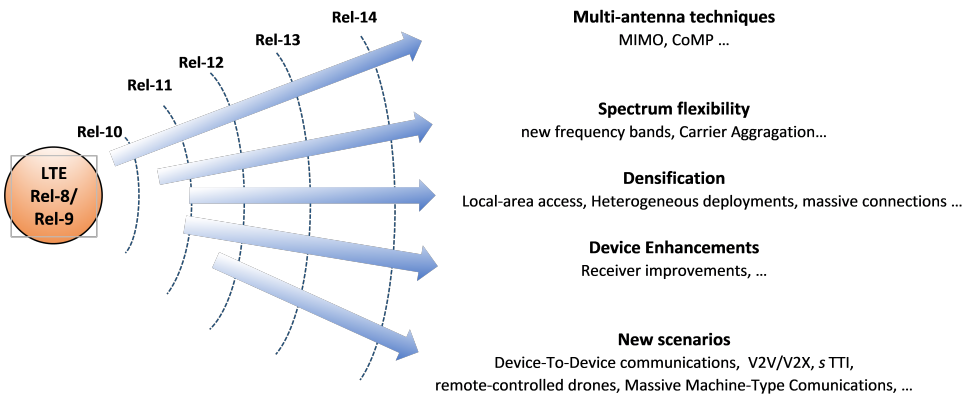


Figure 2.1: LTE evolution
(taken from [5])

LTE evolution starts with **release 10**, completed in late 2010, with the main target of ensuring the fully compliance of the LTE radio-access technology with the IMT-Advanced requirements. This LTE release introduced enhanced LTE spectrum flexibility through **Carrier Aggregation (CA)**: as it can be observed in Figure 2.2, up to five Component Carrier (CC), possibly of different bandwidth, are aggregated and jointly used for transmission to/from a single terminal, allowing for a maximum

transmission bandwidth of 100 MHz.

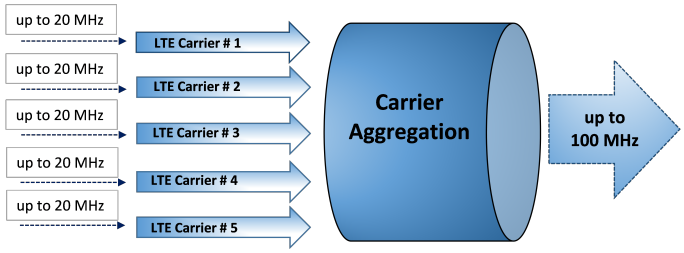


Figure 2.2: Carrier Aggregation
(taken from [5])

Several benefits derive from CA employment:

- higher data rates, since the aggregation of carriers increases spectrum resources;
- capacity gain, thanks to trunking gains from dynamically scheduling traffic across the entire spectrum;
- optimum utilization of an operator's spectrum resources: most of operators are proprietary of fragmented spectrum covering different bands and Carrier Aggregation helps to combine these into a more convenient spectrum resource.

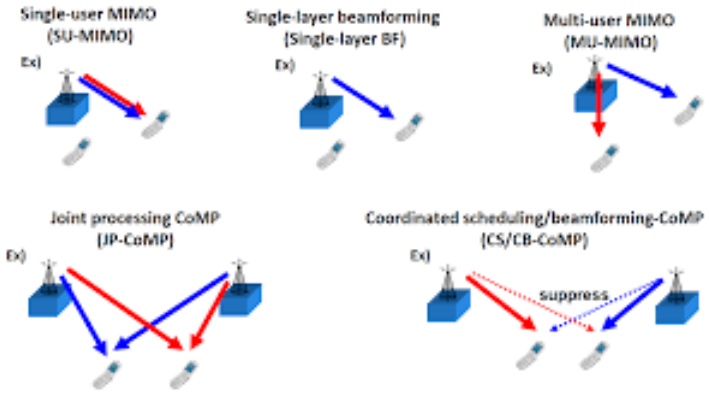


Figure 2.3: COordinated Multi-Point configurations
(taken from [W18])

Release 11 of LTE, completed in late 2012, focused on the so called **Coordinated Multi-Point (CoMP)** transmission and reception: it is a tool to improve the coverage with high data rates and the cell-edge throughput. CoMP approaches, schematized in

Figure 2.3, are particularly beneficial for users at the edge of a cell, since they are typically affected by the interference of the neighboring cells. **Release 12**, completed in 2014, was mainly focused on features related to small cell deployment, such as **dual connectivity**, dynamic Time Division Duplexing (TDD) and small-cell on/off, with the aim of improving the adaptability to the network dynamics. **Release 13**, finalized at the end of 2015, introduced two Machine-type communication options: **Narrowband Internet of Things (NB-IoT)** and **LTE-M**; according to reference [5], it can be seen as an intermediate technology step between 4G and 5G New Radio (NR) air interface. The main characteristics of both these technologies are:

- small bandwidth (200 kHz for NB-IoT and 1.4 MHz for LTE-M);
- extended coverage;
- very low power consumption, battery life above 10 years;
- support of massive connections;
- optimization for cheapest terminal cost.

Even if there are many similarities, LTE-M supports higher data-rates, has a coverage enhancement with respect to standard LTE but not so good as NB-IoT and reduces prices in the modules. **Release 14** was completed during 2017 and it introduced a support for Vehicle-Two-Vehicle (V2V) and Vehicle-to-everything (V2X) communication. **Release 15** appeared in the middle of 2018, with the purpose of significantly reducing latency through the so-called *shortened* Transmission Time Interval (*s*TTI) feature.

In conclusion, all past and future advanced LTE releases aim at going beyond the traditional LTE use cases; however, the need to maintain backward compatibility is a very restrictive limit in a context where requirements constantly evolve [5].

2.1.2 5G history and use cases

With the aim of satisfying new demands, 3GPP started the development of a new radio-access technology, known as **NR**, in order to fully and freely exploit the potential of new technologies. 5G deployment has been phased, since NR is a quite complex technology and it was impossible to standardize all its features in time for the first releases. As it is shown in Figure 2.4, the first references to 5G in the standard can be dated back to the middle of 2016 (phase 1) and the first standard solution for NR appeared at the end of 2017 and it is known as **Early Drop** (release 15): this first release addressed non-standalone hotspots and small cells where the core network is still totally LTE-based and an enhanced-NodeB (e-NodeB) still acts as a master. In June 2018 the first **NR Stand Alone** was completed, where a Next Generation (NG) Core definitely

replaced the LTE Core. This solution marked the end of the first phase, leaving space to the second phase with release 16.

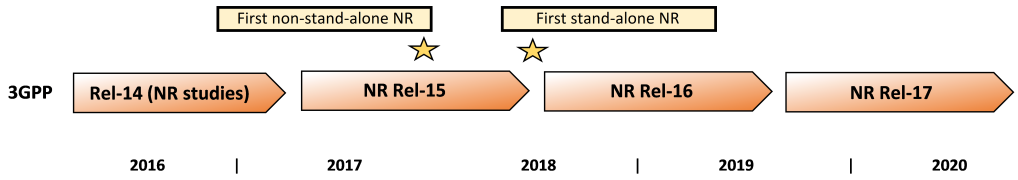


Figure 2.4: 5G time evolution
(taken from [5])

Book [5] provides a clear list of the main benefits introduced by 5G NR with respect to LTE technology:

- supporting very wide transmission bandwidths and the associated high data rates by exploiting higher frequency bands;
- enhanced network energy performance and reduced interference thanks to the adoption of an ultra-lean design;
- low latency to improve performance and enable new use cases;
- extensive usage of beamforming and a massive number of antenna elements not only for data transmission already enabled by LTE, but also for control-plane procedures such as initial access.

In addition to these technology enhancements, a wide range of new use cases characterizes 5G systems, schematized in `Figure 2.5`:

- **Enhanced Mobile BroadBand (EMBB)**, with applications such as virtual/augmented reality, UHD videos, clustering and others scenarios where very high data rates are required;
- **Massive Machine-Type Communication (MMTC)**, with applications such as IoT, Industry 4.0, smart sensors and many other scenarios that foresee a considerable number of connected devices, with few messages per hour to exchange but requiring very low power consumption;
- **Ultra-Reliable Low-Latency Communication (URLLC)**, with applications in the military field, for Connected Cars and for remote surgery.

Release 15 focuses on EMBB with working frequencies below 6 GHz; only from release 16 MMTC and URLLC are considered.

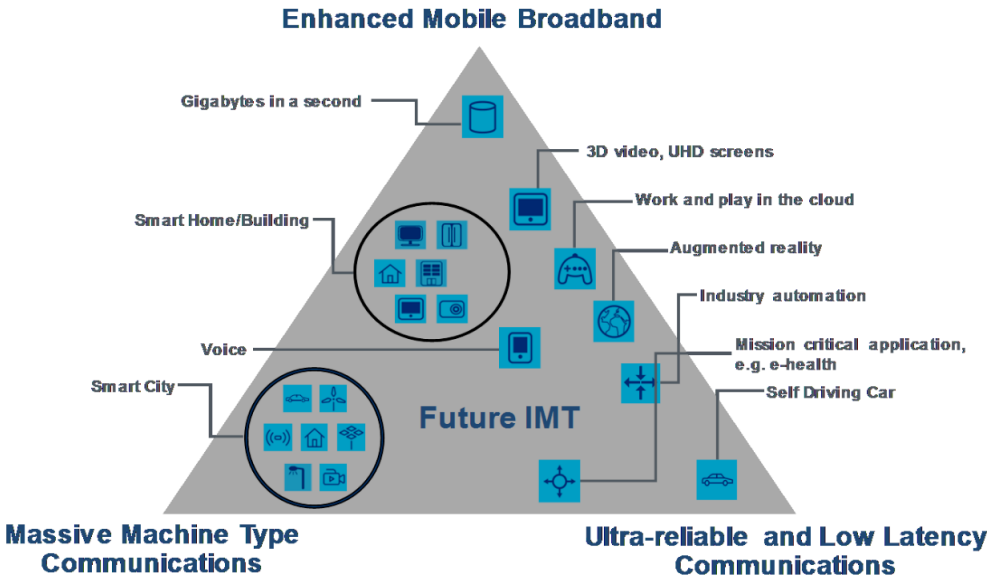


Figure 2.5: 5G use cases
(taken from [W13])

2.2 5G Physical Layer

After discussing about the useful applications of NR, that certainly acted as drivers for the new technology development, it is important to understand which are the key technological enablers behind the 5G revolution:

- **New Radio (NR)**, which includes waveform, numerology, frame structure and physical channels;
- **New bands: millimeter Wave (mmWave)**;
- **Massive Multiple-Input Multiple-Output (Massive MIMO)**;
- **Network slicing**;
- **New Core Network**.

The following subsections explore the first three points of the list, while network slicing and 5G core network are not considered: the idea behind this choice is to keep the focus on the technological changes that have substantially renewed the physical layer of mobile communications.

2.2.1 New Radio

5G NR is a new air interface, that is the radio frequency portion of the circuit between the mobile device and the active base station [W6]. It can be described in terms of **waveform**, **numerology**, **frame structure** and **physical channels**.

NR Waveform

5G technology adopts the same modulation as LTE, **Cyclic Prefix Orthogonal Frequency-Division Multiplexing (CP-OFDM)**, but with three important changes:

- a scalable numerology is introduced;
- both Uplink (UL) and Downlink (DL) work with the same waveform, thus simplifying the overall system design;
- an improvement in terms of spectrum confinement is obtained by performing filtering operations at the transmitter side (i.e. **Filtered Orthogonal Frequency-Division Multiplexing (OFDM)**).

Thanks to the filtering process, it is possible to reduce the guard band, with a consequent improvement in spectrum utilization (up to 98% against the 90% of LTE).

NR numerology

In the context of 3GPP 5G standardization, the concept of **numerology** refers to the configuration of waveform parameters, such as the **Subcarrier Spacing (SS)** ($\Delta f = 2^{\mu} \cdot 15\ kHz$) and the cyclic prefix length (normal or extended), depending on the parameter μ given by the higher layers. Table 2.1 reports some of the possible configurations of NR numerology.

Table 2.1: Standard numerologies for 5G

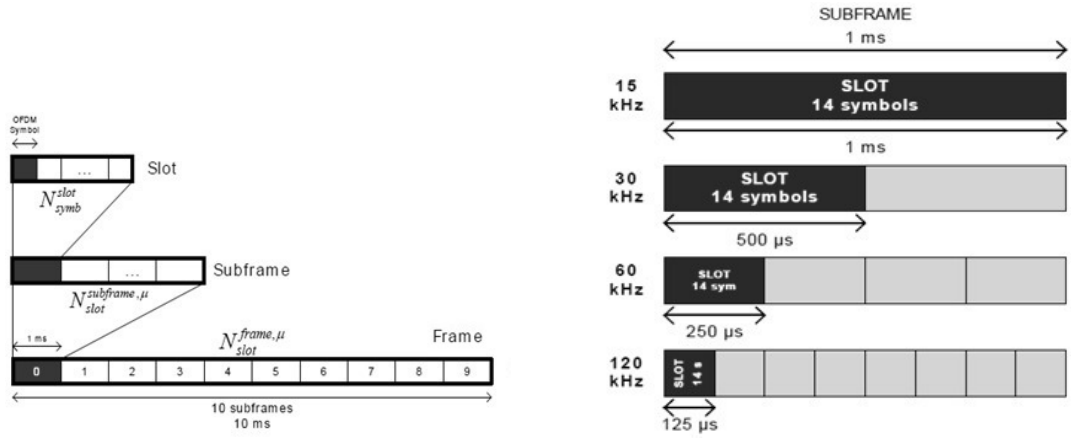
Frequency range [MHz]	μ	Δf [kHz]	**Max bandwidth** [MHz]	**Cyclic prefix** [μs]
450 - 6000	0	15	50	5
	1	30	100	2.5
	2	60	200	1.25/4.17
6000 - 52600	2	60	200	1.25/4.17
	3	120	400	0.62
	4	240	N.A.	0.31

This numerology flexibility can be exploited according to the specific requirements: for example the use of a tight subcarrier spacing is recommended when it is necessary to achieve a good spectral efficiency and a considerable robustness against InterSymbol Interference (ISI); on the contrary, in contexts where low latency and large bandwidth support are required, a larger subcarrier spacing is preferable.

As regards the cyclic prefix, for frequencies below 6 GHz larger values are selected, since cells are quite big and the delay spread can reach a handful of milliseconds. When frequencies are above the 6 GHz threshold, cells have a small size that implies the possibility to handle the delay spread issue with a relatively short cyclic prefix; anyway, the higher frequencies exacerbate the problem of phase noise, forcing to use higher subcarrier spaces (Δf).

Frame structure

The NR time-domain structure consists of a radio **frame** lasting 10 ms, divided into 10 **subframes** of 1 ms each (`Figure 2.6a`).
A subframe is composed by **slots** containing 14 **OFDM symbols** each. The slot duration depends on the numerology: in fact, an OFDM symbol lasts $\frac{1}{\Delta f}$ and consequently a higher subcarrier spacing implies a shorter time slot, as `Figure 2.6b` shows.



(a) NR frame structure

(b) NR subframe structure

Figure 2.6: 5G NR frame structure
(taken from [W23] and [W24])

Nevertheless, the increase of the subcarrier spacing causes the shortening of the cyclic prefix, making this approach not always feasible in scenarios where a low

latency is desired. For this reason, the standard provides an efficient alternative solution for cases with critical latency: each NR slot can be partitioned into **mini-slots**, with flexible starting position and a duration that can correspond to 7,4 or 2 OFDM symbols. Apart from low-latency applications, this flexibility also brings benefits when working in mmWave band: the amount of bandwidth available is abundant, due to the employment of large Δf values, and many packets can be stored into few OFDM symbols [5].

NR can operate both in Frequency Division Duplexing (FDD) and Time Division Duplexing (TDD): at lower frequencies allocations are usually paired, that means FDD, while at higher frequencies TDD is preferable. One key 5G technology component is the so called **dynamic TDD**, defined in reference [5] as the possibility for the dynamic assignment of time-domain resources between the uplink and downlink transmission directions.

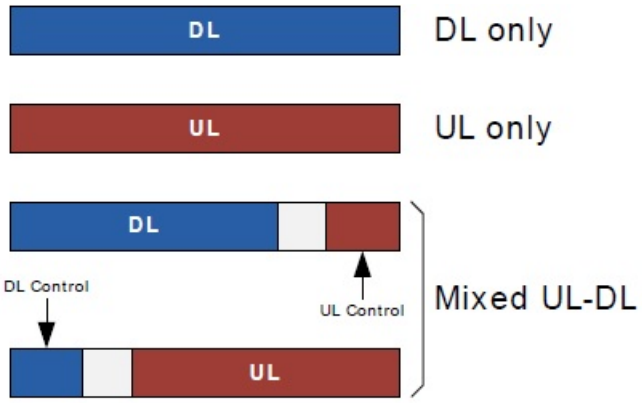


Figure 2.7: NR slot structure
(taken from [W22])

Figure 2.7 shows the possible configuration of NR time slots: they can accommodate only DL symbols, only UL symbols or both of them. In order to distinguish between Downlink, Uplink and Flexible OFDM symbols, the User Equipment (UE) can check the **Slot Format Indicator (SFI)**, that carries an index to a pre-configured table storing the possible link directions for each single slot format.

As regards the frequency domain, the basic scheduling units are the so called resource blocks. A **Resource Block (RB)** is defined as 12 consecutive OFDM subcarriers in the frequency domain, as Figure 2.8 shows.

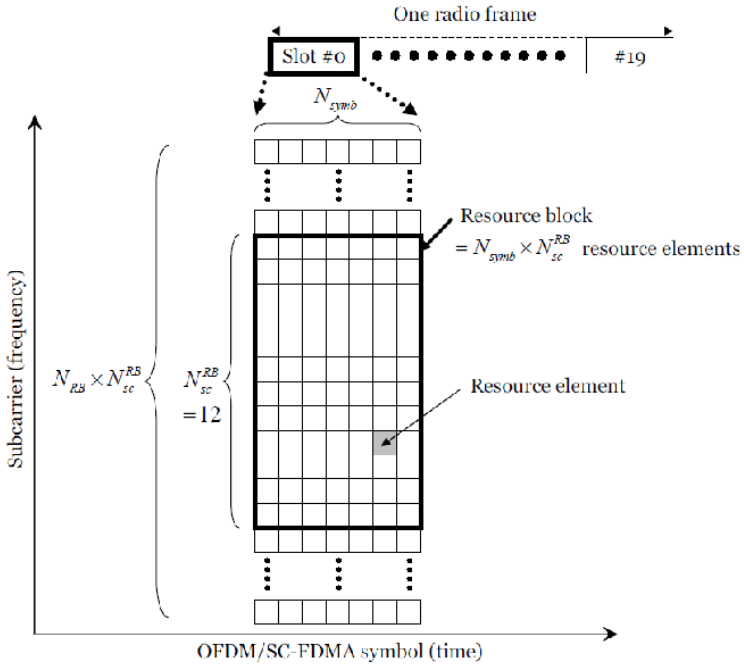


Figure 2.8: NR frequency-time grid
(taken from [W1])

2.2.2 mmWave

Current NR specifications distinguish two frequency ranges in which 5G systems can operate, as `Table 2.2` shows.

Table 2.2: Frequency ranges from 5G specifications

Range denomination	Frequency range
FR1	450 MHz - 6 GHz
FR2	24.25 GHz - 52.6 GHz

Carrier frequencies in the second range belong to the mmWave domain: in fact, the correspondent wavelengths range from 5.7 mm to 12.4 mm, explaining the reason for the name *millimeter* (*mm*). At these frequencies some challenges arise:

- path loss: according to Friis equation, the received power decrease with the square of the frequency that implies 20 dB of additional power loss increasing the carrier frequency by an order of magnitude. Indeed, this is true only if the dimension of the antenna decreases together with the wavelength; maintaining instead the same physical size of the antenna, for example by creating arrays of antenna elements

(i.e. Multiple-Input Multiple-Output (MIMO)), the path loss remains constant with the increase of the frequency;

- blockage: small wavelengths have higher penetration losses and poor diffraction capabilities around obstacles like buildings, humans etc. For this reason, at mmWave frequencies **dual connectivity** is an important feature to counteract this blockage issue: the idea is to ensure control signaling through lower frequency bands and fast fall-back to connectivity options that ensure a good coverage when mmWave connections do not.

2.2.3 Massive MIMO

Massive MIMO is one of the key physical-layer technologies for 5G-based wireless access. The main concept is to increase the number of antennas with respect to the current MIMO systems: standard MIMO systems involve 2 or 4 antennas while massive MIMO technology scales the number of antennas to several tens or even hundreds. In this way very high beamforming gains are achieved and more users can be served in parallel on the same time-frequency resources.

The concept of **beamforming** refers to a signal processing technique that employs **antenna arrays** to provide signal transmissions or receptions with directionality. The introduction of the Active Antenna System (AAS) allows the control of phase and gain through active components characterizing each antenna element. As shown in Figure 2.9, a 2D antenna array enables 3-dimensional (3D) beamforming (i.e. **Full Dimensional MIMO (FD-MIMO)**), since the transmitted radio waves can be controlled on both elevation and azimuth planes, with the additional advantage of keeping the deployment space relatively small.
Massive MIMO offers two most important benefits:

- enhanced coverage and capacity: many terminals can be spatially multiplexed in the same time-frequency resource, increasing the system capacity (**Multi-User Multiple-Input Multiple-Output (MU-MIMO)**); in alternative, beamforming capabilities can be exploited to provide better coverage (**Single-User Multiple-Input Multiple-Output (SU-MIMO)**), better directing the signal towards the target receiver;

- great energy efficiency: the higher gain provided by the introduction of the 2D AAS permits a reduction of radiated power.

However, the real deployment of Massive MIMO brings several challenges:

- dimension and weight of 2-dimensional (2D) planar arrays are considerable, limiting the deployment to high-frequency scenarios;

- the presence of active components for each antenna element gives rise to critical cooling issues.

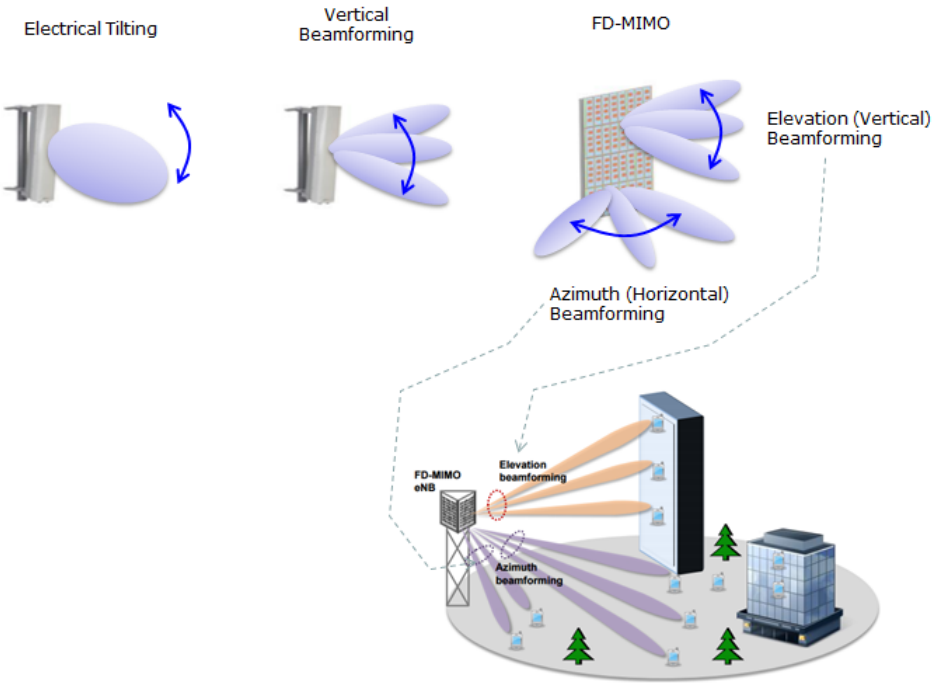


Figure 2.9: 5G FD-MIMO through Massive MIMO (taken from [W10])

Chapter 3

Machine Learning for 5G physical layer

This third chapter deals with the application of machine learning to wireless communication systems. In particular, the focus is on 5G-based technologies: New Radio wireless networks will be expected to support very high data rates and new applications that will open the way for a radically new radio technology paradigm [13].

The authors of paper [13] identify this innovative paradigm in a machine learning-based approach, aiming at satisfying all the diverse requirements of Next-Generation wireless networks by means of an intelligent adaptive learning process. In fact, as remarked in the same paper, the purpose of these new technologies is to learn the colorful characteristics of the system under service and autonomously determine the optimal configurations. With this in mind, future smart mobile terminals are supposed to autonomously access the most fitting spectral bands and to select the appropriate transmission protocol, with the support of efficient machine learning-based algorithms which simultaneously monitor different parameters such as the spectral efficiency, the energy consumption and the quality of service level provided [13].

Even though many studies have been started to explore all the possible ML applications for the upper layers of wireless communication systems, reference [22] asserts that the potential use of Deep Learning to the physical layer has been increasingly recognized; this sounds reasonable thinking about the new features of Next-Generation communications, such as complex scenarios, with a consequent lack of information about the channel model, high data rates and accurate processing requirements.

The following sections summarize a series of attempted applications of Deep Learning algorithms to the physical layer of communication systems; particular

attention is devoted to 5G-based communications, with an emphasis on their peculiar characteristics that make Deep Learning a promising approach.

3.1 Deep Learning for Wireless Physical Layer: an introduction

In recent years, several advanced wireless applications appeared, like **virtual reality**, **augmented reality** and **Internet of things**, to cite the best known examples. As the authors of paper [22] claim, this new scenario propels the advent of 5G technology as a solution for some specific challenges:

- very **high data rates**;
- **low latency**;
- **massive connectivity**.

Chapter 2 describes some enabling technologies that have been introduced in NR systems, such as **massive multi-input-multi-output**, **mmWave** and **ultra-densification network**; however, as reference [22] reports, these technologies present several limitations, especially in complex scenarios:

- difficult channel modeling in complex scenarios: the real channel condition and how it is captured by the mathematical model significantly impact the communication system performance; in complex scenarios, where the channel is full of imperfections and nonlinearities, conventional mathematical models show quite poor performances. This raises the need for systems that are able to complete the communication process without a defined channel model;
- block-wise optimization: conventional communication systems are developed as a series of signal processing blocks (e.g. modulation/demodulation, coding/decoding, detection); by optimizing each block independently, the optimal performance for the whole communication system is not guaranteed. A different approach, aiming at directly optimizing the entire end-to-end performance, needs to be developed.

In this scenario, ML has regained attention for its achievements especially in the upper layers, such as in cognitive radio, positioning and resource management; however, some attempts at ML applications to the physical layer have been carried out, like for example in **modulation recognition**, **encoding** and **decoding**, **channel estimation** and **equalization**, and researchers believe that DL-based systems could replace manual feature extractors, learning features automatically and flexibly adapting the structure and the parameters of the model, with a considerable improvement of the end-to-end performance [22].

References [17] and [22] list some of the main reasons to support the adoption of DL techniques:

- deep networks are very good and adaptable function approximators: they learn the weights of the model optimizing the overall performance through a training process, instead of requiring a rigorous mathematical description that could be hard to extract in complex scenarios;
- DL-based algorithms work very well with large amounts of data, due to their intrinsic nature of parallel and distributed computing, leading to a speeding up of computation;
- DL models can overcome the block structure to have a complete view of the system and optimize it end-to-end.

Before elaborating on some specific DL applications, it is important to highlight this last general concept, clearly expressed in reference [17]: as long as it is possible to design analytic algorithms able to capture real effects, DL will hardly bring significant advantages and indeed it will just add complexity. Nevertheless, DL will be a promising approach to rethink completely the communication system design in complex scenarios where conventional mathematical algorithms struggle.

3.2 Some examples of Deep Learning applications to Wireless Physical Layer

In recent years, much research has been devoted to introduce innovative Deep Learning-based architectures into several processing blocks: the idea is to outperform conventional communication algorithms in the emerging complex scenarios. Some papers mentioned in the references ([17], [22], [9]) illustrate several studies that have produced good results.

3.2.1 Deep Learning for channel estimation

In order to achieve a high-resolution channel estimation, authors of reference [24] have developed a deep neural network for OFDM systems. The network is trained offline under different channel conditions and it outputs the recovered input signals without explicitly estimating the Channel State Information (CSI). In this procedure, the transmitted symbols and the received OFDM signals are fed into the DNN, which is trained to minimize the difference between the input and the output of the network. As paper [24] reports, the obtained results demonstrate the ability of DNNs in learning and

analyzing the characteristics of the wireless channel, especially when non linearities, interference and frequency selectivity are involved. In particular, Deep Learning models turn out to achieve performances that are comparable to traditional algorithms when the number of pilots is large enough, but they often work better when pilots are few or when they are affected by non-linear noise or interference.

3.2.2 Deep Learning for CSI feedback reduction

In paper [23] a deep-learning approach is adopted to achieve CSI reduction and reconstruct the CSI for channel estimation. The DNN system is composed by an encoder and a decoder: the encoder performs a Discrete Fourier Transform (DFT) operation and a linearization of the transformed matrix in a vector with reduced dimension; then the CSI is recovered by the decoder, which consists of a CNN followed by a RefineNet. As the authors of the paper claim, the experiments demonstrate that this DL solution can recover CSI with significantly improved reconstruction quality and effective beamforming gain compared with existing Compressive Sensing (CS)-based methods.

3.2.3 Deep Learning for encoding and decoding

Paper [17] shows how to represent an end-to-end communication system as a DNN autoencoder. An autoencoder is a system that is able to represent the input in a low-dimensional space and to reconstruct it at the output with a little error margin. In the DNN autoencoder, the transmitter is implemented as a feedforward NN with multiple dense layers, followed by a normalization layer. The receiver consists of a feedforward NN but the last dense layer implements a softmax activation, which outputs a probability vector over the set of possible messages. Finally, the channel between transmitter and receiver is modeled by an additive noise layer. This autoencoder is trained by a Stochastic Gradient Descent (SGD) algorithm with cross-entropy as loss function. The authors of the paper state that simulation results show a significant performance improvement with respect to the traditional Hamming code, without requiring any encoder and decoder functions.

3.2.4 Deep Learning for signal classification

Paper [18] presents a data-driven model for Automatic Modulation Classification (AMC), an appealing technique for environment and transmitter identification. The model is based on Long Short-Term Memory (LSTM) and it learns amplitude and phase information to classify 11 typical modulation schemes appearing in the training set.

Performance is analyzed in scenarios with an SNR ranging between 0 and 20 dB: in this context, the model achieves a classification accuracy of 90% on average. In the context of low SNR a Convolutional Neural Network, with its learnable powerful filters, would be a better solution instead.

3.2.5 Deep Learning for MIMO detection

In reference [9], the authors present a Detection Network (DetNet) model, an emerging deep learning-based framework for MIMO detection. The strength of this approach consists in its applicability to various models by performing a single training, with the knowledge, however, that the choice of both the loss function and the structure of the NN deeply influences the system performance. Typical searching algorithms can be employed to perform MIMO detection, but the computational complexity makes them unsuitable in several scenarios; this criticism makes the proposed DetNet an appealing innovative technique. According to reference [9], simulation results prove that it performs better in terms of Bit Error Rate (BER) with respect to several existing works, including the zero forcing, approximate message passing and semidefinite relaxation approaches.

3.3 A focus on Deep Learning for 5G Physical layer

In reference [9], the authors suggest some efficient schemes for Deep Learning-based 5G scenarios. In recent years, several appealing techniques have been developed for 5G communication systems: non-orthogonal multiple access, Massive MIMO, mmWave technologies and many others, as it has been thoroughly discussed in Chapter 2. However, as the authors of [9] claim, these techniques alone are not sufficient to meet all 5G requirements and new communication theories should be established.

Deep learning-based NOMA

Non-Orthogonal Multiple Access (NOMA) is an innovative technique introduced to boost spectral efficiency and system capacity; its performance strongly depends on the CSI, since interference cancellation needs an accurate knowledge of the channel conditions. However, since the channel characteristics in multiple-user systems are very complicated, conventional methods are incapable of capturing changes of the channel conditions in real time. For this reason, Deep Learning-based approaches have been attempted [8].

In the same paper, a framework that integrates LSTM and NOMA is presented; the LSTM is a special type of Recurrent Neural Network (RNN) architecture, which

is able to keep in memory values over intervals of arbitrary length. It is extended with additional hidden layers, in order to improve learning and representation capabilities, at the same time keeping the training set quite small. The proposed LSTM network is composed of 6 hidden layers followed by Rectified Linear Unit (ReLU) layers, one input layer processed by a Restricted Boltzmann Machine (RBM) and one output layer processed by a sigmoid function instead.

The authors of the paper conclude that further research on NOMA should consider the integration with DL, to take advantage from its powerful learning ability. Other researchers indeed have followed this suggestion: in reference [9] the whole developed NOMA system is regarded as a blackbox and a DNN is employed for approximating the whole NOMA system, consisting in the Base Station (BS), the wireless channels, all the users, etc..

Deep Learning-based massive MIMO

Massive MIMO could potentially achieve very high gains, but requires a perfect CSI knowledge; this requirement has propelled the investigation on Deep Learning performance for both high-resolution channel and direction of arrivals (DOA) estimation and CSI reconstruction.

The approach proposed in reference [11] achieves end-to-end learning by modeling the whole system as a DNN, which empowers the performance of the Direction of Arrival (DOA) estimation. The channel matrix is determined by the DOA information and the complex gain:

- the DNN framework for high-resolution DOA estimation consists of an input layer, an encoder, a decoder and an output layer. The encoder is composed of two hidden layers to which a ReLU activation function is applied: their role is to learn and encode the features of the signal coming from the input layer; then a dropout layer is adopted to counteract the overfitting issue. The decoder instead includes a hidden layer with a ReLU activation step right after;

- for complex gain estimation the DNN learning framework for DOA estimation is exploited; then, the noise contribution estimate is subtracted to the estimated output signal, which can be decomposed to obtain the gain matrix.

Therefore, exploiting the results obtained in the DOA estimation process, the channel estimation step is formulated after completing the proposed complex gain estimation method. According to the authors, simulation results have demonstrated that the proposed scheme can achieve better performance in terms of both DOA estimation and tracking and high-resolution channel estimation, compared with traditional methods.

Deep Learning-based mmWave techniques

mmWave communication scenarios are affected by ultra-high power consumption and limited gains. To overcome these limitations, several papers, such as references [10] and [9], propose a deep-learning mmWave Massive MIMO framework for effective hybrid precoding, in which the selection of the precoders to achieve the optimal decoder is regarded as a mapping relation in the deep neural network. Experiments on these approaches have shown excellent performance in terms of Bit Error Rate and spectrum efficiency.

Chapter 4

A Deep Learning approach for channel estimation in 5G systems

Channel estimation is a challenging problem in wireless systems: the transmitted information is subject to highly random distorting effects, such as reflection, scattering, diffraction, that considerably limit the performance of the system; moreover, the mobility of transmitters and receivers causes rapid changes in the channel response over time. All these factors makes the channel estimator a very sensitive block of every wireless communication system.

The first section of this fourth chapter is dedicated to **DeModulation-Reference Signals**, introduced in NR to estimate the radio channel at the receiver side for the associated physical channel demodulation: the purpose is to clearly outline the new 5G-framework in which dedicated reference signals are used specifically for coherent demodulation, highlighting their properties and their standardized configurations.

Given these reference signals, there exist several conventional pilot-based estimation techniques that exploit the known-values in the time-frequency grid, corresponding to the pilots, to derive all the unknown values of the channel response. These algorithms transform the estimation task into an optimization problem, aiming at minimizing a certain objective function:

- Least Squares (LS) estimator minimizes the sum of the squared differences between the observed values and the fitted values provided by the model;
- Minimum Mean Square Error (MMSE) estimator minimizes the mean square error of the fitted values. In general, its performance are better with respect to LS, since it exploits the statistics of the channel and noise variance;
- Approximated Linear Minimum Mean Square Error (ALMMSE) estimator is an

approximated linear version of the MMSE that reduces the size of the correlation and filtering matrices.

An overview on the first two techniques is provided in the second section, with a particular attention devoted to the low-rank 2D-MMSE Wiener filter adopted in the New Radio simulation platform developed in Telecom Italia laboratories. To enhance these traditional approaches many studies have supported the introduction of Deep Learning-based solutions in the channel estimation process.

The rest of the chapter instead focuses on the application of Deep learning for channel estimation in mobile communication systems: the third section analyses the solution proposed in paper [21] for LTE-based systems, where the time-frequency grid is treated as a 2D-image which is known only at the pilot positions; finally, in the fourth section this Deep Learning-based solution is revisited and adapted to 5G scenarios, with the final purpose of evaluating its performances compared to a more traditional approach, illustrated in `Subsection 4.2.2`.

4.1 Demodulation Reference Signals

DeModulation-Reference Signal (DM-RS) are transmitted in downlink and uplink to enable coherent demodulation at the receiver side. In reference [5], the authors highlight their most relevant characteristics:

- front-loaded design, enabling low latency transmissions: locating the reference signals in the first symbols of the transmission, the receiver can compute the channel estimate earlier, thus demodulating the received symbols on the fly, without waiting for the end of the slot;
- support for up to 12 orthogonal ports for MIMO scenarios, by combining Code Division Multiplexing (CDM) and Frequency Division Multiplexing (FDM) techniques;
- support for flexible transmission duration, from 2 to 14 symbols;
- configurations with up to four resource elements per slot dedicated to DM-RS, enabling the support of very high-speed scenarios.

4.1.1 DM-RS properties

As the authors of [5] highlight, one of the most important properties of DM-RS is a well-focused time-domain autocorrelation: with this aim in mind, $2^{31} - 1$-length Gold sequences are used for OFDM-based modulation. Gold codes are a family of binary sequences with good correlation properties: in particular, their normalized autocorrelation function can assume one of the following four values:

$$\{1, \frac{1}{p}, \frac{-\beta(N)}{p}, \frac{\beta(N)-2}{p}\} \tag{4.1}$$

where $\beta(N) = 2^{\frac{\lfloor N+2 \rfloor}{2}}$ and $p = 2^N - 1$ is the length of the Gold sequence. Similarly, the cross-correlation function can assume the values

$$\{\frac{1}{p}, \frac{-\beta(N)}{p}, \frac{\beta(N)-2}{p}\} \tag{4.2}$$

So, for any pair of Gold sequences in the same family, i.e. generated from the same pair of m-sequences, the cross-correlation function presents low values, allowing to deal with more users in the system: in fact, these codes are used on top of pseudo-random sequences to assure orthogonality in Code Division Multiple Access (CDMA) scenarios. The underlying pseudo-random sequences are generated across all the Common

Resource Blocks (CRBs) in the frequency-domain but transmitted only in the resource blocks allocated for data transmission, where channel estimation is needed [5]. As the authors remark, the reason for generating the Reference Signals (RS) sequences across all the resource blocks is to guarantee a perfect alignment between the underlying sequences that are used for devices scheduled on overlapping time-frequency resources in MU-MIMO scenarios: if this condition is not fulfilled, reference signals for different co-scheduled users do not satisfy the orthogonality property.

4.1.2 Time-domain configurations of DM-RS

The standard includes two different time-domain structures, characterized by a different location of the first DM-RS symbol [5]:

- **mapping Type A**: the first DM-RS is located in the OFDM symbol $l_0 = 2$ or $l_0 = 3$ within the time slot. This type of mapping targets the cases where transmitted data occupy most of the slot;
- **mapping Type B**: the first DM-RS is located in the first OFDM symbol allocated to data. It is important to observe that, differently from the previous mapping scenario, in this case the DM-RS starting position is not given relatively to the slot boundaries, but relatively to data location. This kind of mapping is intended for short transmissions, occupying a small portion of the time slot and enabling very low latency transmissions.

Although the use of front-loaded reference signals successfully meets the low-latency requirements of NR realities, another problem needs to be solved: when the channel varies rapidly, such as in high-speed scenarios, the front-loaded configuration alone is not able to capture the variations characterizing the channel. To support these challenging scenarios, the NR standard includes the possibility of configuring up to three additional DM-RS resources in a single time slot: the receiver can exploit the additional DM-RS occasions for the coherent channel estimation, performing an interpolation operation between the distinct DM-RS points and improving, in this way, the estimation quality. It is worth noting that, differently from what happens in LTE, NR standard does not allow interslot interpolation, since different slots could be allocated to different transmissions, with different destinations and different beam directions [5].

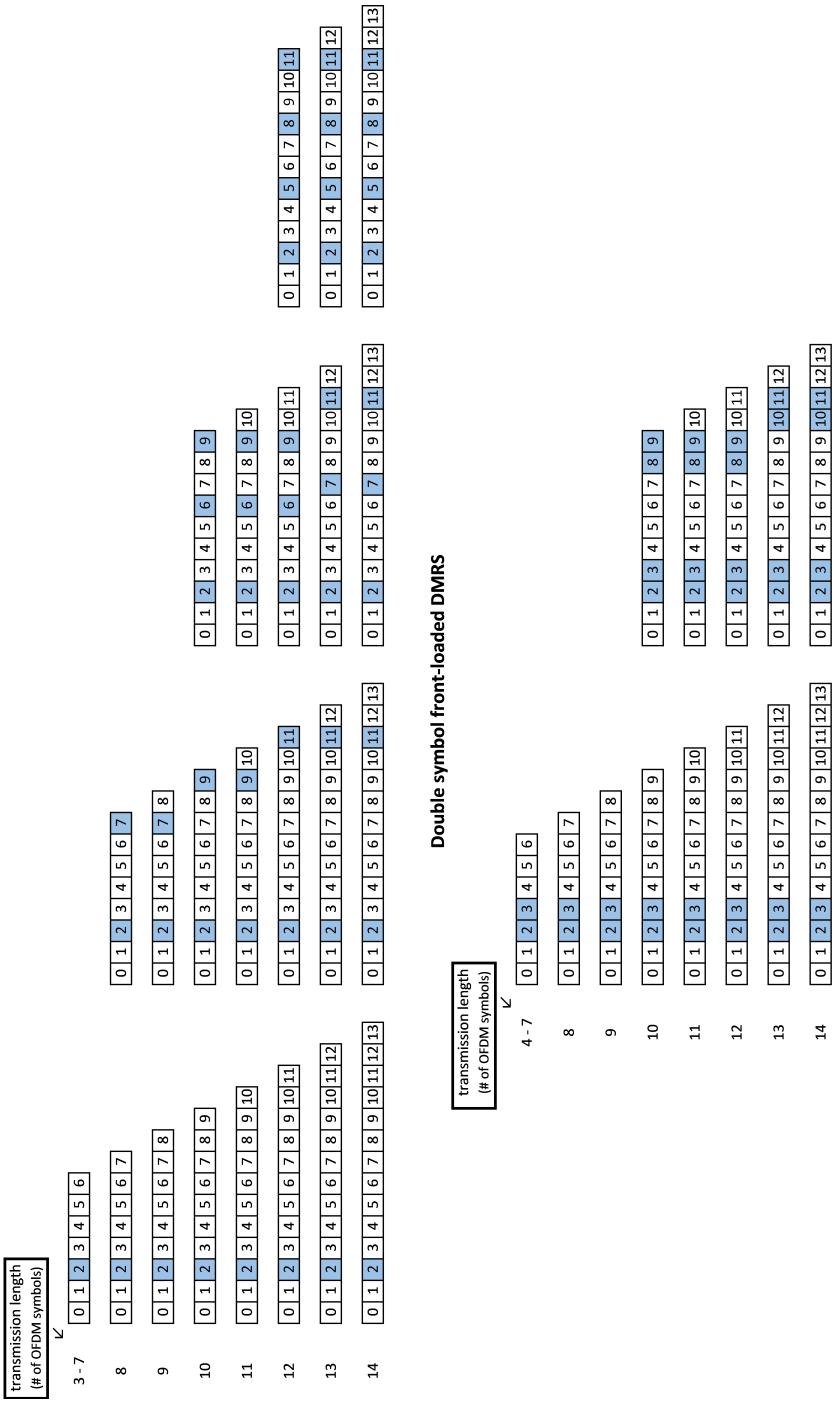


Figure 4.1: Time-domain allocations for DM-RS mapping Type A

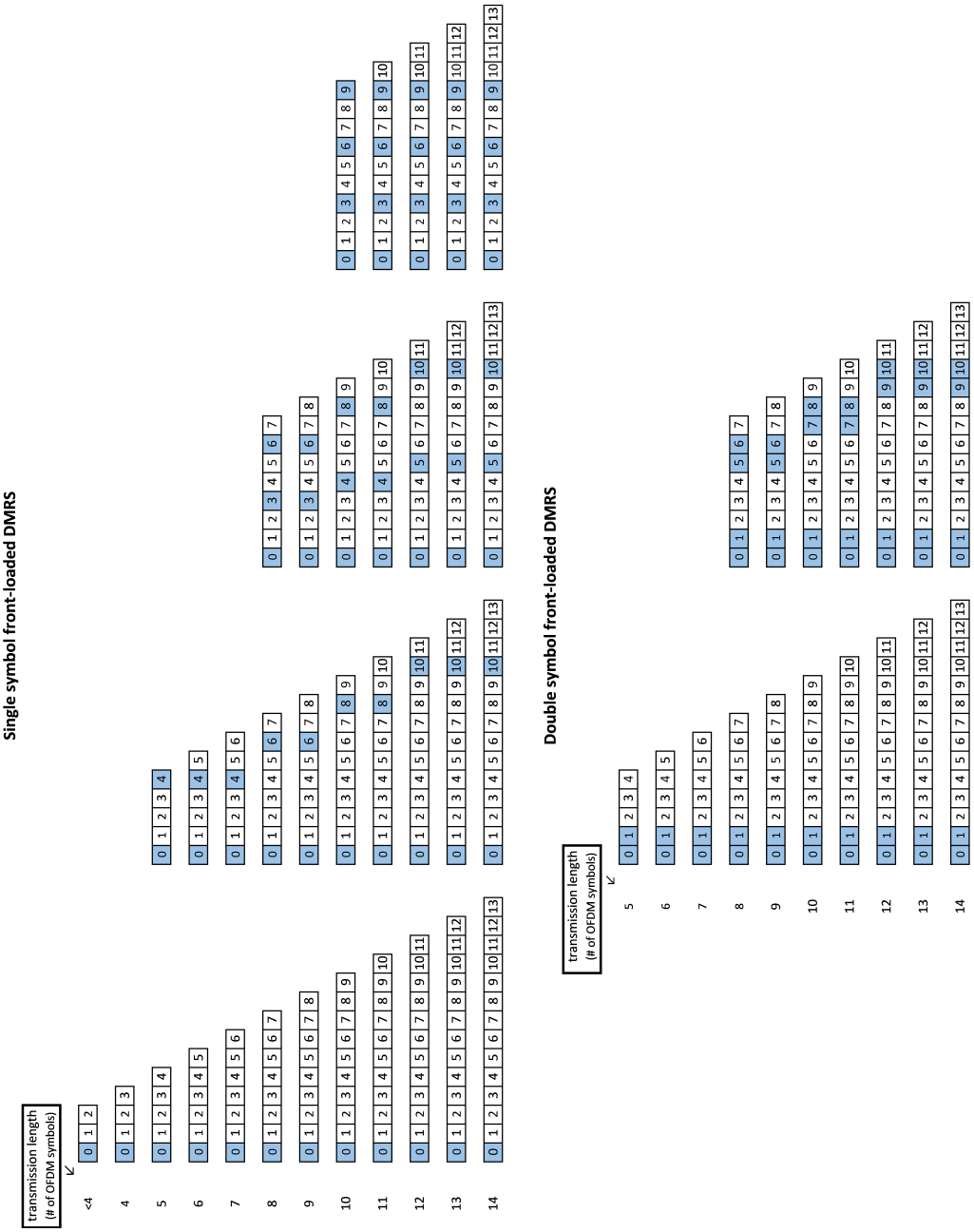


Figure 4.2: Time-domain allocations for DM-RS mapping Type B

The standard includes different allocation schemes for DM-RS, with the possibility of introducing additional DM-RS symbols (double-symbol configuration) basically to increase the number of scheduled antenna ports. In order to separate reference signals related to different antenna ports, a combination of different multiplexing techniques is exploited: Code Division Multiplexing (CDM), Frquency Division Multiplexing (FDM) and, in case of double-symbol configuration, Time Division Multiplexing (TDM). Basing on DM-RS allocation in the frequency-domain and on the maximum number of configurable orthogonal reference signals, two types of DM-RS can be distinguished, **DM-RS Type 1** and **DM-RS Type 2**.

DM-RS Type 1

As Figure 4.3 shows, this type of DM-RS can provide up to four orthogonal signals in single-symbol configuration and up to eight orthogonal signals in double-symbol configuration. In single-symbol configuration, antenna ports 0 and 2 are allocated on even-numbered subcarriers in the frequency domain and they are multiplexed in code-division (CDM), by multiplying the underlying pseudo random sequence with different orthogonal sequences of length 2 in the frequency domain: in this way, reference signals corresponding to antenna port 0 turns to be orthogonal to the ones corresponding to antenna port 2. With the same logic, antenna port 1 and 3 use the same subcarriers (odd-numbered) but different length-2 orthogonal sequences, resulting in a code-domain separation. In general, antenna ports that are allocated on the same subcarriers and are separated in code-domain form a **Code Division Multiplexing group (CDM group)**; this means that antenna ports are separated in the code-domain within the CDM group and in the frequency-domain between CDM groups.

Double-symbol configuration is adopted when more than four orthogonal antenna ports are needed: as it is clarified in reference [5], two consecutive OFDM symbols are allocated, so that a length-2 orthogonal sequence can be used to extend the code-domain separation to also include the time domain, doubling the total number of available orthogonal sequences with respect to single-symbol configuration.

DM-RS Type 2

These type of reference signals present some differences with respect to the DM-RS Type 1, as Figure 4.4 shows: first of all, each CDM group consists of two pairs of two consecutive subcarriers, instead of one as in Type 1; then, over the pair of subcarriers an orthogonal sequence of length 2 is used to multiplex in the code-domain all the antenna ports belonging to the same CDM group. As a consequence of this, DM-RS Type 2 can provide up to six orthogonal signals in single-symbol configuration and up to

twelve orthogonal signals in double-symbol configuration; in fact, since each resource block contains 12 subcarriers, a maximum of three CDM groups can be obtained using one single OFDM symbol, resulting in a maximum of six orthogonal antenna ports. If instead a second symbol is allocated and a 2-length sequence is introduced in the time-domain, the maximum number of orthogonal antenna ports is doubled.

The authors of book [5] provide a remarkable observation: while Type 1 is denser in the frequency domain, Type 2 trades the frequency-domain density for a larger multiplexing capacity, that means a larger number of orthogonal reference signals. This makes DM-RS Type 2 more suitable to support MU-MIMO scenarios with simultaneous transmission to a large number of devices.

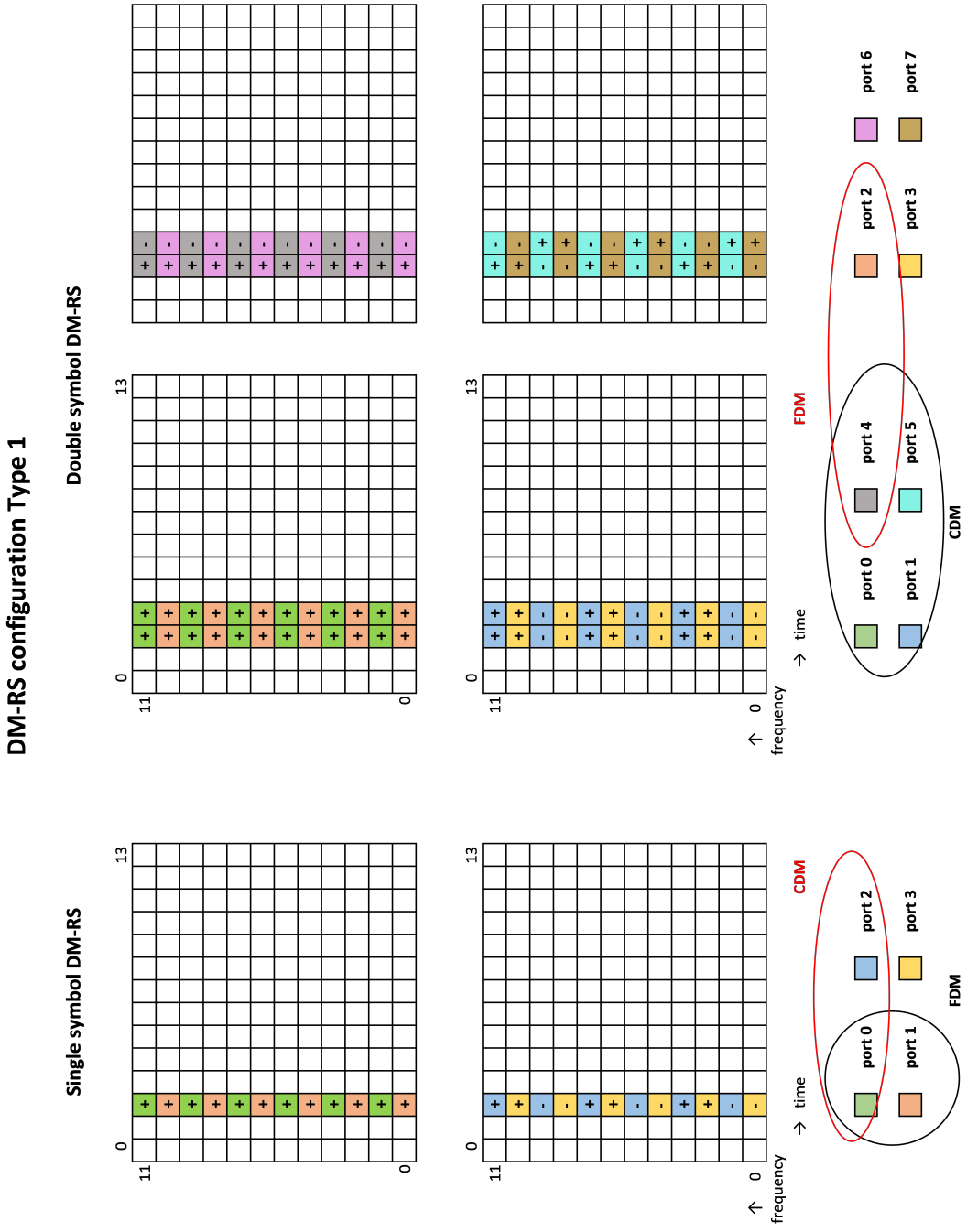


Figure 4.3: DM-RS Type 1

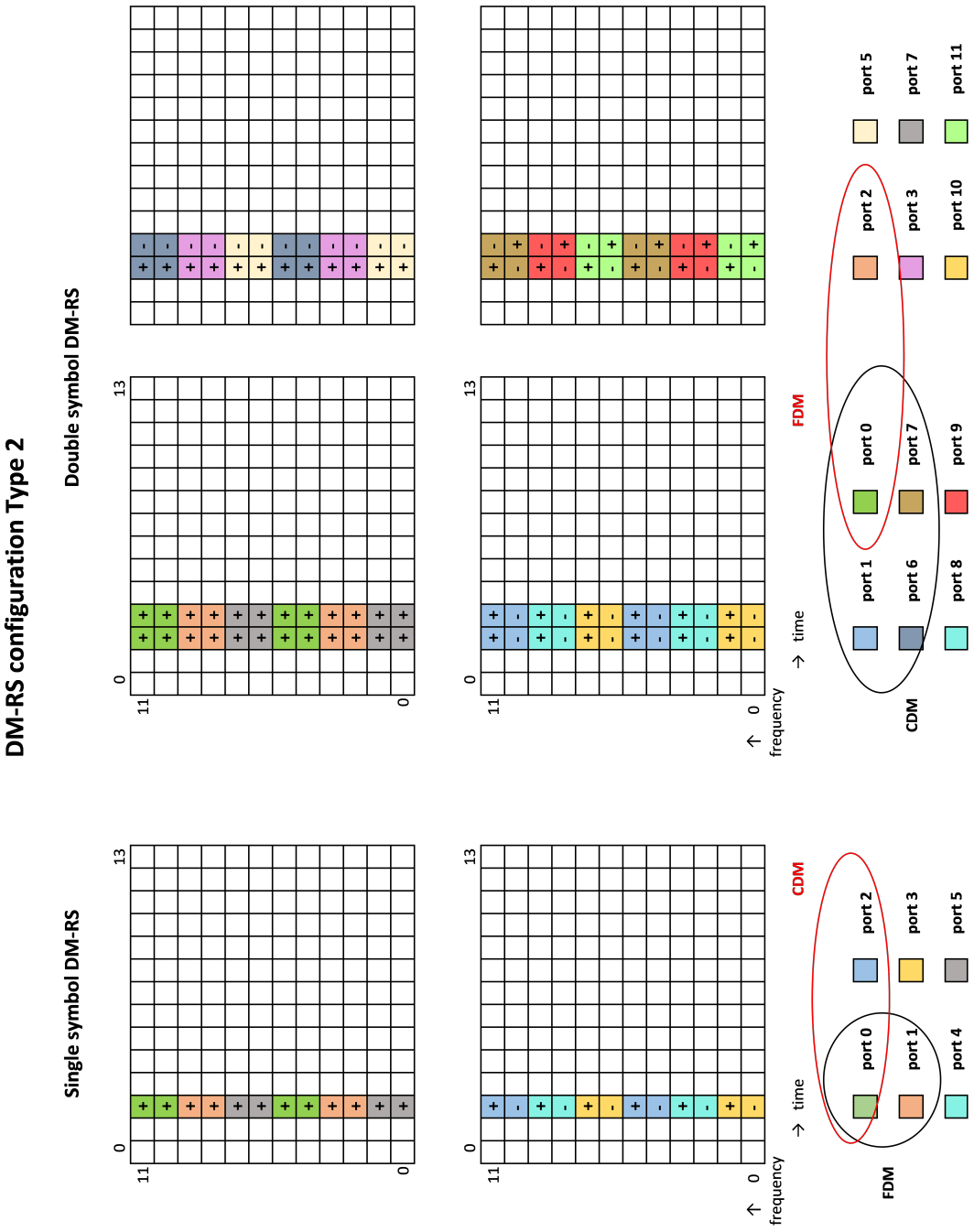


Figure 4.4: DM-RS Type 2

4.1.3 Multiplexing between DM-RS and data

The reference signal structure is configured based on a combination of dynamic scheduling decisions, communicated to the intended device through the Downlink Channel Information (DCI), and higher-layer configuration [5]. The DCI carries information about:

- the DM-RS ports assigned to the device;
- the number of consecutive symbol configured for DM-RS (single-symbol or double-single configuration);
- the presence of co-scheduled DM-RS CDM groups for other devices in MU-MIMO scenarios: this information allows the device to map the data around both its own reference signals and the reference signals allocated for other devices. As a consequence, both a good degree of dynamism in changing the number of co-scheduled users and the exploitation of the resource elements of unused CDM groups for Physical Downlink Shared CHannel (PDSCH) data transmission are enabled;
- the existence of multiple spatially-multiplexed layers for the same device in SU-MIMO scenarios: to avoid interlayer interference for the DM-RS, each layer is prevented from occupying resource elements intended to another CDM group of the same device;
- the number of DM-RS CDM groups not used for PDSCH data transmission, used to determine the number of resource elements reserved for DM-RS.

DCI format 1_1 is used for the scheduling of PDSCH in one cell: it contains a field of 4 bits for DM-RS port assignment. The table in Figure 4.5 reports the relation between the number of CDM groups and DM-RS ports reserved for reference signals and the value assumed by the field in the DCI, for DM-RS Type 1: the number of CDM groups without data can assume values 1, 2, and 3, which refers to CDM groups {0}, {0,1}, and {0, 1, 2} respectively, while the column of DM-RS ports reports the indexes of the antenna ports assigned to the device.

One Codeword: Codeword 0 enabled, Codeword 1 disabled				Two Codewords: Codeword 0 enabled, Codeword 1 enabled			
Value	Number of DMRS CDM group(s) without data	DMRS port(s)	Number of front-load symbols	Value	Number of DMRS CDM group(s) without data	DMRS port(s)	Number of front-load symbols
0	1	0	1	0	2	0-4	2
1	1	1	1	1	2	0,1,2,3,4,6	2
2	1	0,1	1	2	2	0,1,2,3,4,5,6	2
3	2	0	1	3	2	0,1,2,3,4,5,6,7	2
4	2	1	1	4-31	reserved	reserved	reserved
5	2	2	1				
6	2	3	1				
7	2	0,1	1				
8	2	2,3	1				
9	2	0-2	1				
10	2	0-3	1				
11	2	0,2	1				
12	2	0	2				
13	2	1	2				
14	2	2	2				
15	2	3	2				
16	2	4	2				
17	2	5	2				
18	2	6	2				
19	2	7	2				
20	2	0,1	2				
21	2	2,3	2				
22	2	4,5	2				
23	2	6,7	2				
24	2	0,4	2				
25	2	2,6	2				
26	2	0,1,4	2				
27	2	2,3,6	2				
28	2	0,1,4,5	2				
29	2	2,3,6,7	2				
30	2	0,2,4,6	2				
31	Reserved	Reserved	Reserved				

Figure 4.5: DM-RS port assignment through DCI format1_1 field for Type 1 (taken from [2])

4.2 Pilot-based channel estimation algorithms for wireless OFDM systems

In coherent OFDM systems, channel estimation is a fundamental block of the receiver design. Before reaching the receiver side, information passes through the radio channel, which distorts the transmitted data: in order to correctly demodulate the useful signal, the effect of the wireless channel must be estimated and compensated.

Pilot-based channel estimation techniques are based on the multiplexing of pilots symbols (i.e. known symbols) into the transmitted data: these symbols are scattered on the time-frequency grid and the channel attenuation in neighboring positions can be recovered through a two-dimensional interpolation.

4.2.1 Channel estimators: notation and general features

Paper [20] identifies two main classes of estimators:

- **two-dimensional estimators**: the complexity of these estimators is usually quite large;
- **separable estimators**: to reduce the complexity in multidimensional signal processing, separable filters are often adopted.

Since these types of estimators are linear, it sounds reasonable to compare their complexity on the basis of the average number of multiplications per estimated attenuation [20].

Before exploring these different solutions, it is necessary to introduce the simple notation adopted by the author of the paper. The channel attenuation values estimated at pilot positions can be denoted as

$$p_{k,l} = \frac{y_{k,l}}{x_{k,l}} \tag{4.3}$$

where $y_{k,l}$ is the received signal at subcarrier k in the l-th OFDM symbol, while $x_{k,l}$ is the corresponding transmitted pilot. $h_{k,l}$ indicate the final estimates of the channel attenuation values, each one computed as the linear combination of a set of $p_{k,l}$; the set of pilots which contribute to the estimation is determined by the particular estimator used.

The minimum mean-squared error estimator of $\boldsymbol{h}$ is

$$\hat{\boldsymbol{h}} = \boldsymbol{R}_{\boldsymbol{hp}} \boldsymbol{R}_{\boldsymbol{pp}}^{-\mathbf{1}} \boldsymbol{p} \tag{4.4}$$

where the vector $\boldsymbol{h}$ contains the attenuation values that must be estimated, the vector $\boldsymbol{p}$ contains the channel estimates at pilot positions, $\boldsymbol{R}_{\boldsymbol{hp}}$ is the cross-covariance matrix between $\boldsymbol{p}$ and $\boldsymbol{h}$ and $\boldsymbol{R}_{\boldsymbol{pp}}$ is the auto-covariance matrix of $\boldsymbol{p}$.

Separable filters: Least Squares technique

Since 2D filters are generally characterized by large computational complexity, a good trade-off between performance and complexity could be represented by the outer product of two separate 1-dimensional (1D) filters in the time and frequency directions, with a significantly reduction in the number of multiplication per used pilot [20]: basically, after a 1D filter is applied in the frequency direction, a 1D filter is applied in the time direction to complete the interpolation over the whole time-frequency grid.

Time Domain Least Squares algorithm, implemented in the LTE link level simulation platform developed in Telecom Italia laboratories, is an example of channel estimation technique based on separable filters. In fact, with LS, the interpolation is done independently in frequency and time: a first interpolation between pilot subcarriers in frequency domain is followed by a linear interpolation over subsequent OFDM symbols. If we consider an OFDM baseband system, where $X(k)$ and $Y(k)$ are the transmitted and received symbols in frequency-domain respectively, $g(\tau)$ is the Channel Impulse Response (CIR) and $w(t)$ is sum of thermal noise and interference, the received signal in frequency-domain can be expressed as

$$\boldsymbol{Y} = \boldsymbol{XFg} + \boldsymbol{W} \tag{4.5}$$

where $\boldsymbol{X}$ is a matrix with the transmitted symbols $X(0), X(1), ..., X(N-1)$ on the main diagonal, $\boldsymbol{F}$ is the DFT matrix of size $N \times N$ and $\boldsymbol{H} = \boldsymbol{Fg}$ is the frequency response Channel Frequency Response (CFR) of the wireless channel.

The LS procedure includes several steps, performed separately for each OFDM symbol containing pilots:

- received pilot symbols are compensated in frequency-domain and converted to the time-domain by means of an Inverse Fast Fourier Transform (IFFT) operation;
- the resulting CIR is then filtered by a matrix $\boldsymbol{Q_{LS}}$;
- the interpolated CFR ($\hat{H}(n,k)$) is finally obtained applying an Fast Fourier Transform (FFT) operation to the filtered CIR.

At this point, the output consists of an estimate of the channel frequency response computed on each frequency subchannel but only over the OFDM symbols that carry some pilots: this means that a further linear interpolation is performed in order to estimate the CFR also in the OFDM symbols without pilots. `Figure 4.6` summarizes the estimation process.

The filtering function is expressed by

$$\hat{\boldsymbol{H}} = \boldsymbol{FQ_{LS}F^H X_p^H Y} \tag{4.6}$$

The filtering matrix $\boldsymbol{Q_{LS}}$ is computed as

$$\boldsymbol{Q_{LS}} = (\boldsymbol{T^H X_p^H X_p T})^{-1} \tag{4.7}$$

where $\boldsymbol{T}$ consists of the first L columns of the DFT matrix $\boldsymbol{F}$: this means that CIR filtering considers only the first L taps with larger energy and excludes low energy taps of $g(\tau)$, with a consequent improvement of Mean Square Error (MSE) performance.

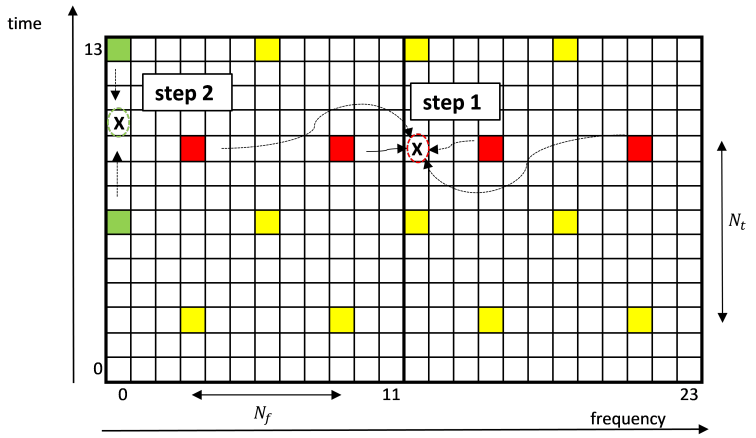


Figure 4.6: Time Domain LS estimation
(taken from internal)

2D filters estimators: 2D-MMSE Wiener filter

In OFDM systems, the optimal linear channel estimator is the one that minimizes the MSE between the estimated CFR and the original one. It is possible to demonstrate that the optimal linear estimator is a **2D-MMSE Wiener filter**, which combines the set of K surrounding pilots to estimate the channel attenuation in a specific position on the time-frequency grid, where K is the maximum number of multiplications imposed by complexity requirements; indeed, this estimation algorithm basically consists of a two-dimensional convolution operation. `Figure 4.7` shows an example of how the $K = 4$ nearest pilots (in red) are selected for the estimation of an attenuation value (**X**).

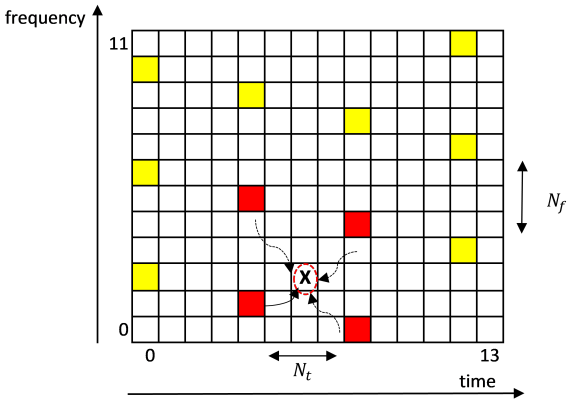


Figure 4.7: 2D FIR Wiener filter applied to an LTE signal
(taken from internal paper)

At the receiver side, the CFR is known only in correspondence of the pilots: since for coherent demodulation the whole channel attenuation must be computed, the remaining $\hat{H}(n,k)$ have to be estimated by interpolating the known $\hat{H}(n_p,k_p)$ coefficients:

$$\hat{H}(n,k) = \sum_{(n_p,k_p)\in P} w^{n,k}_{n_p,k_p} \hat{H}(n_p,k_p) \tag{4.8}$$

where P is the set of the positions that accommodate the nearest pilots with respect to the position (n,k). The filter coefficients $w^{n,k}_{n_p,k_p}$ are computed taking into account the autocorrelation function $R_{HH}(\Delta n,\Delta k) = E\{H(n,k)H^*(n-\Delta n,k-\Delta k)\}$: the idea is that the more a pilot is far from the position to estimate, the less it contributes to the estimation, since the correlation between the pilot symbol and the estimated symbol is low.

To derive the solution to the equation that determines the weights of the 2D-MMSE filter, it is necessary to define some variables:

- $\boldsymbol{p}$ is a vector of size $L \times 1$ containing the L pilot symbols which are taken into account; L is generally smaller than the total number of transmitted pilots, to reduce the complexity of the estimation;
- $\hat{\boldsymbol{h}}$ is a vector of size $M \times 1$ where M represents the number of attenuation values that have to be estimated;
- $\boldsymbol{R_{hp}} = E\{\boldsymbol{h}\boldsymbol{p}^{\boldsymbol{H}}\}$ is the cross-covariance matrix between the CFR estimated in positions (n,k), without pilot, and the CFR estimated in pilot positions (n_p,k_p);
- $\boldsymbol{R_{pp}} = E\{\boldsymbol{p}\boldsymbol{p}^{\boldsymbol{H}}\}$ is the auto-covariance matrix of $\boldsymbol{p}$.

The solution of the equation can be expressed as

$$\hat{\boldsymbol{h}} = \boldsymbol{R_{hp}}\boldsymbol{R_{pp}^{-1}}\boldsymbol{p} \tag{4.9}$$

Matrix $\boldsymbol{R_{hp}}$ can be calculated as

$$R_{hp}(i,j) = E\{H_{n,k}H^*_{n_p,k_p}\} = R_t(n-n_p)R_f(k-k_p) \tag{4.10}$$

where $= R_t(n-n_p)$ and $R_f(k-k_p)$ are the autocorrelation of the CFR in time and frequency domain respectively. In the same way, matrix $\boldsymbol{R_{hp}}$ can be calculated as

$$R_{pp}(i,j) = E\{H_{n_{pa},k_{pa}}H^*_{n_{pb},k_{pb}}\} = R_t(n_{pa}-n_{pb})R_f(k_{pa}-k_{pb}) + \sigma_n^2\delta(n_{pa}-n_{pb},k_{pa}-k_{pb}) \tag{4.11}$$

where the i-th element of $\boldsymbol{p}$ is in position (n_{pa},k_{pa}) and the j-th element is in position (n_{pb},k_{pb}). The term $\sigma_n^2\delta(n_{pa}-n_{pb},k_{pa}-k_{pb})$ represents the noise plus interference

contribution, added on the diagonal of the matrix $\boldsymbol{R}_{pp}$.

It is important to observe that matrices $\boldsymbol{R}_{pp}$ and $\boldsymbol{R}_{hp}$ can be computed just once for different delay spreads and different doppler frequencies and so the complexity of the algorithm results decreased.

4.2.2 2D-MMSE Wiener filter with reduced complexity

The general version of the 2D-MMSE estimator is typically too complex to be actually implemented: for this reason, the NR link level simulation platform developed in Telecom Italia laboratories adopts a simplified 2D Wiener filter. To reduce the complexity of the original algorithm, it is possible to implement a **low-rank 2D-MMSE filter**, that reduces the rank of the 2D filter by reducing the set P: in other words, only the nearest pilots are used to estimate a certain value of the channel attenuation. In practice, this can be realized by means of a **sliding window** that includes only the surrounding pilot symbols in the grid. Clearly, the window size determines the accuracy of the estimation: a smaller window implies a significant reduction of the algorithm complexity, at the cost of a coarser accuracy. The appropriate window size depends on the channel characteristics, in particular on the channel delay spread: in case of a high delay spread, channel a small window has to be chosen, since the channel varies fast and thus the correlation between a certain attenuation value and the surrounding pilots decreases faster with the distance; on the contrary, when the channel varies slowly, the contribution of further pilots may be still important for the estimation. For implementation reasons, some limitations are imposed to the choice of the window size:

- the window size in the frequency-domain must be an odd number of Physical Resource Blocks (PRBs);
- the window size in the time-domain is fixed to 1 Transmission Time Interval (TTI).

`Figure 4.8` shows an example of low-rank 2D-MMSE filter with sliding window size equal to 3.

The most complex operation of the algorithm is the inversion of the autocorrelation matrix $\boldsymbol{R}_{pp}$. To solve this complexity issue, the exact inverse matrix computation can be substituted by an iterative procedure which exploits the Conjugate Gradient method: with this alternative, complexity is significantly reduced but a loss of accuracy occurs, making this approach suitable only for high Signal to Noise plus Interference Ratio (SINR) scenarios. This low-rank 2D-MMSE algorithm is used as a reference point for the performance evaluation of the Deep Learning-based channel estimator that we have developed and that will be described in the following sections.

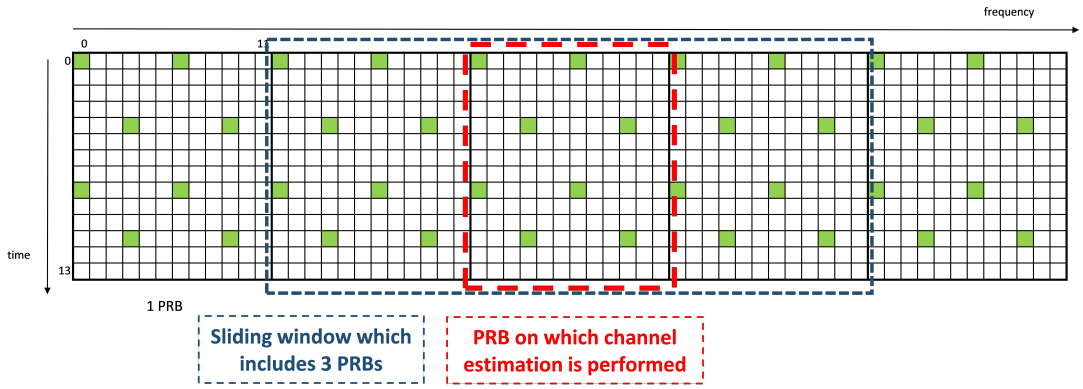


Figure 4.8: Low-rank 2D-MMSE Wiener filter
(taken from internal paper)

4.3 Deep Learning-based channel estimation

To enhance the traditional estimators described in the previous section, many researches have proposed the introduction of Deep Learning-based algorithms in the channel estimation process. Among all the studies on this topic, one of the most interesting and complete is certainly presented in paper [21] and will be analyzed in this section.

The idea of the authors is to treat the time-frequency grid of the channel response as a Low Resolution (LR) 2D-image, whose pixels are known only at the pilot positions. A Deep Learning-based approach is then adopted, consisting of two different phases:

- an **image Super-Resolution (SR)** algorithm, that enhances the resolution of the LR input image and transforms it into an High Resolution (HR), by estimating the channel response values at all positions without pilots;

- an **Image Restoration (IR)** algorithm, that removes the noise effect.

For SR and IR implementation, two recently developed Convolutional Neural Networks are used:

- **Super-Resolution Convolutional Neural Network (SRCNN)**: this network is proposed in paper [7] with the purpose of mapping a LR into an HR image;

- **Denoising Convolutional Neural Network (DnCNN)**: this network is proposed in paper [25] to remove or reduce the noise affecting an image.

4.3.1 Image Super-Resolution Convolutional Neural Network

Paper [7] proposes a deep CNN that takes a low-resolution image as the input and outputs the high-resolution version. Furthermore, the authors establish a relationship between the Deep Learning-based technique and the traditional Sparse-Coding (SC)-based method. A typical SC algorithm involves overlapping patches cropped from the input image that are pre-processed and encoded through a low-resolution dictionary, obtaining sparse coefficients. Then, a high-resolution dictionary maps these coefficients into high-resolution patches that produce the final output by means of a weighted averaging operation. This series of steps are equivalent to a deep convolutional neural network, named **Super-Resolution Convolutional Neural Network** in which:

- patch extraction and aggregation are formulated as convolutional layers;
- dictionaries are not explicitly learned, since they are implicitly achieved through hidden layers.

The only pre-processing operation foreseen is an interpolation of the input low-resolution image, in order to upscale it to the desired size. In reference [7], the authors denote the interpolated low-resolution image with $\boldsymbol{Y}$, the ground truth high-resolution image with $\boldsymbol{X}$ and the mapping operation with F. The mapping procedure conceptually consists of three steps, described in [7]:

1. patch extraction and representation: several patches are extracted form the low-resolution image and mapped onto high-dimensional vectors, whose number of dimensions corresponds to the number of feature maps collected;
2. non-linear mapping: the high-dimensional vectors output by the previous step are mapped onto other high-dimensional vectors through a non-linear operation. The resulting vectors are representations of high-resolution patches and store another set of feature maps;
3. reconstruction: in this last phase, the final high-resolution image is generated aggregating the high-resolution patch-wise representations.

Patch extraction and representation

This first step consists in extracting patches and representing them through a set of bases such as PCA or Discrete Cosine Transform (DCT), which means convolving the image by a set of filters. The authors of the paper formulate this first layer (F_1) as

$$F_1(\boldsymbol{Y}) = max(0, W_1 * \boldsymbol{Y} + B_1) \tag{4.12}$$

where W_1 corresponds to n_1 filters of support $c \times f_1 \times f_1$, being c the number of channels of the input image and f_1 the spatial size of the filter, and B_1 consists of an n_1-dimensional vector whose elements are associated with the filters. Summarizing, W_1 applies n_1 convolution operations (indicated by "$*$") on the input image, each of kernel size equal to the support of the filters. At this point, the output for each extracted patch consists of n_1 feature maps, to which a ReLU operation is performed.

Non-linear mapping

In the second step, each of the n_1-dimensional feature vectors is mapped into an n_2-dimensional one, equivalent to applying n_2 filters with support 1×1. In paper [7], this second layer is formalized as follow:

$$F_2(\boldsymbol{Y}) = max(0, W_2 * F_1(\boldsymbol{Y}) + B_2) \tag{4.13}$$

where W_2 consists of n_2 filters of size $n_1 \times f_2 \times f_2$, while B_2 is a n_2-dimensional vector. Each of the n_2-dimensional output vectors is nothing but a high-resolution representation of a patch.

Reconstruction

The predicted overlapping high-resolution patches are typically averaged to obtain the final output image [7]; this averaging operation is equivalent to a convolution by a predefined filter on a set of feature maps. For this reason, the third and last layer can be expressed as

$$F_3(\boldsymbol{Y}) = W_3 * F_2(\boldsymbol{Y}) + B_3 \tag{4.14}$$

where W_3 represents c filters of size $n_2 \times f_3 \times f_3$ and B_3 is a c-dimensional vector.

Table 4.1: SRCNN hyperparameters typical setting

f_1	f_2	f_3	n_1	n_2
9	1	5	64	32

These three operative layers are put together to build a convolutional neural network. As regards the hyperparameters setting, the authors propose a typical and basic setting collected in `Table 4.1`. Some remarks made by the authors of [7] can be useful to understand this configuration: choosing the filter size of the last layer (f_3) smaller than that of the first layer (f_1), the model relies more on the central part of the high-resolution patch; choosing $n_2 < n_1$ sounds reasonable since the

n_2-dimensional high-resolution patch representations are expected to be sparser than the n_1-dimensional outputs of the first layer.

Training process

Learning the end-to-end mapping function F consists in learning the network parameters $\boldsymbol{\Theta}_S = \{W_1, W_2, W_3, B_1, B_2, B_3\}$: this is essentially an optimization problem, whose aim is to minimize the loss between the reconstructed images $F(\boldsymbol{Y_i}; \boldsymbol{\Theta})$ and the corresponding true high-resolution images $\boldsymbol{X_i}$. The loss function used for the model proposed in paper [21] is the MSE, computed as

$$L(\boldsymbol{\Theta}_S) = \frac{1}{n} \sum_{i=1}^{n} ||F(\boldsymbol{Y_i}; \boldsymbol{\Theta}) - \boldsymbol{X_i}||^2 \tag{4.15}$$

where n is the number of training samples. In training process, this loss function is minimized through the SGD algorithm with standard backpropagation.

4.3.2 Image Denoising Convolutional Neural Network

The model proposed in paper [25], referred to as **Denoising Convolutional Neural Network**, is able to handle Gaussian denoising with unknown noise level; it isolates the noise in the noisy input image by means of a feed-forward convolutional network. As explained by the authors, this network is designed to predict not the denoised image $\hat{\boldsymbol{x}}$ but the residual image $\hat{\boldsymbol{v}}$, defined as the difference between the noisy observation and the latent clean image: in other words, in the hidden layers of the network the latent clean image is implicitly removed. Also batch normalization is introduced to enhance the training performance, since residual learning and batch normalization can benefit from each other, with a consequent speeding up in training and an improvement of denoising performances [25].

Residual Learning

Residual learning is a technique that faces the problem of accuracy degradation with the increasing of the network depth: in fact, since the residual mapping is much easier to be learned with respect to the original mapping, extremely deep CNNs can be easily trained [25]. As the authors of the paper point out, without residual learning, the input intensity and the convolutional features are correlated with their neighbored ones and the distribution of the layer inputs also depends on the content of the images in each training batch. With the use of residual learning instead, which implies the removal of the latent clean image, the inputs at each layer are characterized by a Gaussian

distribution, a lower correlation and a negligible relationship with the image content [25].

Batch Normalization

Batch normalization is an efficient solution to the internal covariate shift, i.e. changes in the distributions of internal non-linearity inputs during training [25]. The covariate shift is a consequence of the mini-batch gradient descent: mini-batch SGD is a trade-off between Stochastic Gradient Descent (SGD) and Batch Gradient Descent (BGD) [W17], since the cost function, and therefore the gradient, are averaged over a small number of samples (around 10-500), while the SGD batch size is equal to 1 sample and the BGD size corresponds to all the training samples [W7] . As the authors of reference [25] state, batch normalization allows for fast training, better performances and lower sensitivity to initialization.

A DnCNN with depth D is characterized by three different types of layers:

- the first layer, consisting of a convolutional layer followed by a ReLU: this layer generates 64 feature maps from 64 filters of size $3 \times 3 \times c$, where c is the number of image channels; the ReLU is then employed for nonlinearity;

- layers from 2 to $D-1$, consisting each of a convolutional layer followed by a batch normalization layer and a ReLU: these layers adopt 64 filters of size $3 \times 3 \times 64$;

- the last layer, consisting simply of a convolutional layer with c filters of size $3 \times 3 \times 64$.

Training process

The input of the DnCNN is a noisy image $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{v}$ and the network aims at learning the mapping function $R(\boldsymbol{Y}) = \boldsymbol{v}$ that predict the residual image with the minimum error. Since $\boldsymbol{x} = \boldsymbol{y} - R(\boldsymbol{y})$, the loss function selected for the training process is the MSE between the desired residual images and the estimated ones, formulated in [25] as

$$L(\boldsymbol{\Theta}) = \frac{1}{2N} \sum_{i=1}^{N} ||R(\boldsymbol{y}; \boldsymbol{\Theta}) - (\boldsymbol{y_i} - \boldsymbol{x_i})||_F^2 \tag{4.16}$$

where $\{(\boldsymbol{y_i}, \boldsymbol{x_i})\}_{i=1}^{N}$ are the N noisy-clean training image patch pairs and $\boldsymbol{\Theta}$ represents the set of learnable parameters.

4.3.3 ChannelNet for LTE

The authors of paper [21] propose a pipeline for DL-based channel estimation, called ChannelNet, with the purpose of estimating the time-frequency response matrix $\boldsymbol{H}$ characterizing the link between a single transmitter and a single receiver antenna (i.e. Single-Input Single-Output (SISO) channel). Since the matrix $\boldsymbol{H}$ has complex values, it is represented as two 2D-images, one for the real values and one for the imaginary values. The values estimated at the pilot locations $\hat{\boldsymbol{h}_p}$ are considered as the LR and noisy version of the channel image which must be mapped to the HR version:

- first, an SRCNN network takes as input the low-resolution interpolated images (real and imaginary part) and estimates the unknown values of the channel response matrix $\boldsymbol{H}$;
- then, a DnCNN implementation, cascaded with the SRCNN, removes the noise from the estimated images.

Training process

The authors of the paper denote the set of trainable parameters as $\boldsymbol{\Theta} = \{\boldsymbol{\Theta_S}, \boldsymbol{\Theta_D}\}$, where $\boldsymbol{\Theta_S}$ and $\boldsymbol{\Theta_D}$ represent the sets of parameters relative to the SRCNN and the DnCNN respectively. The input to the ChannelNet is represented by the pilot values vector $\hat{\boldsymbol{h}_p}$, that is interpolated and then passed to the SRCNN input layer. The final output instead is denoted as $\hat{\boldsymbol{H}}$ and it can be expressed as

$$\hat{\boldsymbol{H}} = f(\boldsymbol{\Theta}; \hat{\boldsymbol{h}_p}) = f_D(f_S(\boldsymbol{\Theta_S}; \hat{\boldsymbol{h}_p}); \boldsymbol{\Theta_D}) \tag{4.17}$$

where f_S and f_D represent the image Super-Resolution and Image Reconstruction functions, respectively.

The total loss function is the MSE between the estimated and actual channel response, computed as follows:

$$C = \frac{1}{||T||} \sum_{\hat{\boldsymbol{h}_p} \in T} ||f(\boldsymbol{\Theta}; \hat{\boldsymbol{h}_p}) - \boldsymbol{H}||^2 \tag{4.18}$$

where $||T||$ corresponds to the size of the training set. Actually, the authors simplify the training process dividing it in two stages:

- in the first stage, the loss of the SRCNN network is minimized:

$$C_1 = \frac{1}{||T||} \sum_{\hat{\boldsymbol{h}_p} \in T} ||\boldsymbol{Z} - \boldsymbol{H}||^2 \tag{4.19}$$

being $\boldsymbol{Z} = f_S(\boldsymbol{\Theta_S}; \hat{\boldsymbol{h}_p})$ the output of the SRCNN network.

- in the second stage, the weights learned by the SRCNN are considered fixed while the parameters of the DnCNN network are trained by minimizing the loss function C_2 expressed as:

$$C_2 = \frac{1}{||T||} \sum_{\hat{\boldsymbol{h}_p} \in T} ||\hat{\boldsymbol{H}} - \boldsymbol{H}||^2 \tag{4.20}$$

where $\hat{\boldsymbol{H}} = f_D(\boldsymbol{Z}; \boldsymbol{\Theta_D})$ is the output of the DnCNN network, while $\boldsymbol{Z}$ is the output of the SRCNN and the input of the DnCNN.

In paper [21], some observations are reported about the channel conditions in which the training process is performed: through their experiments, the authors determine that the set of learned parameters $\boldsymbol{\Theta}$ is dependent on the Signal-to-Noise Ratio (SNR) value. In other words, the ChannelNet should be re-trained each time the SNR value changes but, being the SNR a continuous value, this approach would be clearly impracticable. Anyway, experiments show that good performances are achieved as long as the SNR varies in a small range of values.

Simulation results

For model simulations, the authors of reference [21] consider a SISO channel configuration, evaluating the MSE over a range of SNR values. For channel modeling and pilot transmission, the LTE simulator developed by the university of Vienna, named Vienna LTE-Advanced (LTE-A) simulator, is used. The DL-based scheme is implemented using Tensorflow and Keras.

Table 4.2: ChannelNet hyperparameters setting

Hyperparameter	Configured value
learning rate	0.001
batch size	128
iterations for SRCNN	300
iterations for DnCNN	200
training samples	32000
validation samples	4000
testing samples	4000

For both SRCNN and DnCNN networks the configured hyperparameters are reported in Table 4.2. Based on LTE standard, simulation frames consist of 14 time

slots and 72 subcarriers. As regards the wireless channel model, Vehicular-A (VehA) and SUI5, which is a model with long delay spread, are considered, with carrier frequency 2.1 GHz, bandwidth equal to 1.6 MHz and user speed of 50 km/h.

For the VehA channel model, the authors compare the accuracy of channel estimation obtained with the proposed DL-based method and the accuracy obtained with three traditional estimators:

- ideal MMSE: clearly this algorithm shows the best performance and represents a lower bound in terms of MSE; however it requires a full knowledge of the channel statistics and this is not a realistic assumption in practical applications;
- estimated MMSE: this algorithm instead estimates the channel statistics (in particular the correlation matrix) based on the received signals;
- ideal ALMMSE: still assuming a full knowledge of channel statistics, this algorithm tries to approximate the ideal MMSE technique.

For their experiments, the authors train one ChannelNet model at SNR equal to 12 dB (denoted by deep low-SNR) and one model at SNR equal to 22 dB (denoted by deep high-SNR). Then, they divide the SNR range into two regions:

- low-SNR region: in this case the channel is estimated adopting the deep low-SNR network;
- high-SNR region: in this case the channel is estimated adopting the deep high-SNR network.

When SNR goes beyond 22 dB, the deep high-SNR performances start degrading and a third trained model would be necessary. Although, at SNR values below 20 dB, simulation results show that ChannelNet models achieve performances comparable with the ideal MMSE but performs better than the ideal ALMMSE and estimated MMSE.

As regards the SUI5 channel model, which is definitely more complex than the VehA model, the situation is different: for SNR values above 5 dB, ideal ALMMSE and estimated MMSE perform really poorly, while the proposed ChannelNet model still achieves acceptable results.

In short, simulation results presented in paper [21] demonstrate that DL-based techniques are highly competitive instruments for channel estimation purposes. For this reason, in `Section 4.5` we will investigate a DL-based solution, inspired by the ChannelNet model just illustrated, but moving to 5G scenarios. However, before describing our experiments and commenting the results obtained, it is fundamental to specify the framework in which simulations have been performed.

4.4 NR Link Simulator: a focus on the channel model

For all the experiments conducted and described in this thesis, a software NR Link Simulator is used. This simulator has been developed in Telecom Italia laboratories with an engine implemented in MATLAB® and some blocks, that are particularly critical in terms of execution speed, implemented in C language and linked to the MATLAB engine via MEX. The purpose is to model a radio interface which is compliant with 3GPP specification and to evaluate the link level performances of 5G-based point-to-point communications; as it is a link layer simulator, it is limited to scenarios with a single NR base station (gNodeB) and a single UE, even if multiple antennas can be involved (i.e. SU-MIMO).

The key element of this simulation framework is clearly the channel model: for a NR-compliant link simulator, the 3GPP RAN WG4 group proposes two options [1]:

- **Tapped Delay Line (TDL)**: for this channel model, the correlation between different antennas is defined statically by a correlation matrix. The TDL model is based on the description of the impulse response of the channel;

- **Cluster Delay Line (CDL)**: with respect to TDL, this channel model allows a better representation of beamforming, since the direction of the signal in the space is modeled; in fact, the model is based on the description of the main departure and arrival directions of the signal in the space and the number of clusters corresponds to the number of channel reflections. However, the antenna correlation is not explicitly defined, indeed it depends on the array geometry and on the channel spread.

The CDL channel model is adopted for all the trials presented in this thesis and for this reason it can be useful to dedicate the following subsection to a detailed description of the model.

4.4.1 CDL channel model

The CDL model is a combination of M Non-Line Of Sight (NLOS) rays and one Line Of Sight (LOS) ray if present; all these rays are defined through a 3D **geometric description** based on angles (θ, ϕ) of arrival and departure.

One important feature of the model is that the channel has to maintain the same unitary power of the input signal: this is done by applying a normalization factor β, which guarantees the same output power as measured in input. The normalization scheme is illustrated in Figure 4.9.

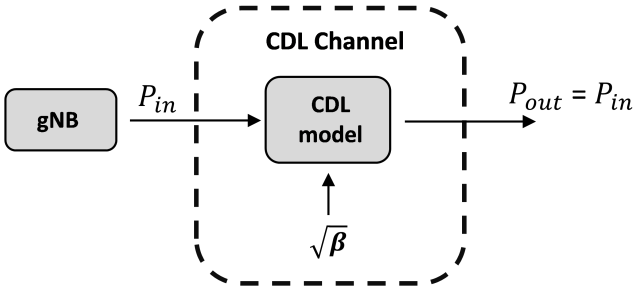


Figure 4.9: Power normalization for CDL channel model in link simulator

Geometric description

The geometric description of the model requires the definition of both the position in space and the geometry of the gNodeB and UE antenna arrays. In Telecom Italia NR link simulator, the gNodeB is equipped with a 2D antenna, while the UE is provided with a linear antenna array. All this information is derived from two *.txt* files: *bsElements.txt* and *ueElements.txt* for gNodeB and UE antenna systems respectively. `Figure 4.10` provides an example of the *.txt* files structure and shows the corresponding spatial configuration of the antenna systems.

MIMO 8x4 configuration	
bsElements.txt	***ueElements.txt***
(0,0,0)	(0,0,0)
(0,0,1)	(0,1,0)
(0,1,0)	(0,0,0)
(0,1,1)	(0,1,0)
(0,0,0)	
(0,0,1)	
(0,1,0)	
(0,1,1)	

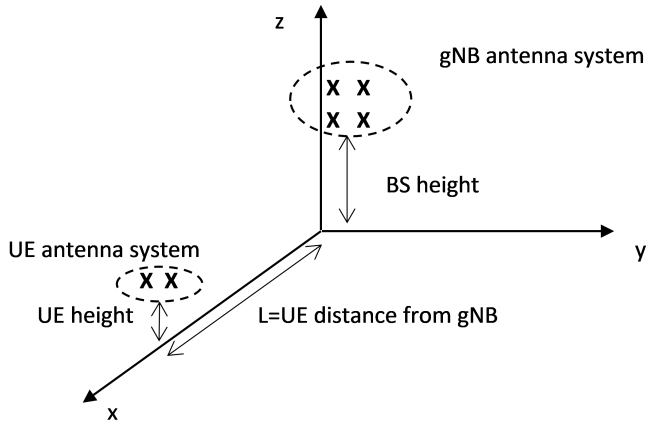


Figure 4.10: Example of *bsElements.txt* and *ueElements.txt* structure

As it is possible to observe in the figure, the UE linear array is placed along the y-axis, while the gNodeB is located in the xz-plane. Moreover, it is important to specify that each antenna element is dual-polarized so that in a 8×4 MIMO configuration only 4 antenna elements in transmission and 2 antenna elements in reception are required.

Model parameters

The parameters fixed by the model are:

- the element distance on both the horizontal and the vertical plane. By default, it is set to half of the wavelength (0.5λ); when increasing or reducing it, the system correlation is proportionally increased or reduced;
- the UE height with respect to the xy-plane;
- the gNodeB height with respect to the xy-plane;
- the main angle of arrival in azimuth and elevation;
- the specific channel profile, since five different CDL channel types are defined;
- the delay profile, which determines the scaling factor for the time of arrival of each cluster signal, this way creating a shorter or longer delay spread;
- the correlation type, that reduces or increases the angle spread at the gNodeB or at the UE;
- the UE travel direction, which determines the direction in which the terminal is moving with respect to the direction of arrival of the signal and thus impacts on the Doppler effect;
- the UE travel speed, that sets the speed at which the terminal is moving and impacts on the channel correlation in time-domain;

Among the five different types of CDL supported, three are specific for NLOS transmissions:

- **CDL-A**;
- **CDL-B**: this profile can be used for sub 6 GHz frequency, where LOS is not mandatory. A characterizing property is that the first path in time is also the strongest one;
- **CDL-C**.

The other two are instead adopted for LOS transmissions:

- **CDL-D**: this profile can be used for frequencies above 6 GHz, where LOS is mandatory. As for CDL-B, the first path in time is also the strongest one;

- **CDL-E**.

For all the experiments that will be presented later, the CDL-B profile is selected, with the working frequency set to 3.64 GHz.

Also regarding the **delay spread profile**, five different types are supported by the CDL channel model:

- very short;
- short;
- nominal;
- long;
- very long.

The simulation results presented in the following pages are obtained by configuring a nominal delay spread.

Three types of correlation for angle departure spread are supported both at the gNodeB and at the UE side:

- low;
- medium;
- high.

For each CDL model, the model defines the Angle Spread (AS) values in azimuth and elevation, measured in [1] and reported in `Table 4.3`.

Table 4.3: Angle Spread values for CDL profiles

CDL model	**AS$_{\text{model}}$** [deg]			
	ASD	ASA	ZSD	ZSA
CDL-A	73.7	85.3	28.6	21.1
CDL-B	41.6	59.3	6.0	10.4
CDL-C	39.1	71.1	4.1	10.4
CDL-D	19.0	21.1	3.0	1.9
CDL-E	13.2	37.6	1.5	2.5

ASD and ASA represent the azimuth angle spread of departure and arrival respectively, while ZSD and ZSA concern the elevation plane. These angle direction

values can be modified during the simulations according to the equation below:

$$\Phi_{n,scaled} = \frac{AS_{desired}}{AS_{model}}(\Phi_{n,model}, \mu_{\Phi,model}) + \mu_{\Phi,desired} \tag{4.21}$$

where $AS_{desired}$ represents the desired values of angle direction, according to the configured correlation profile.

At this point, there are some important observations to make:

- the correlation of the channel strongly depends on the CDL profile chosen: CDL-B profile shows better un-correlation performance with respect to CDL-D;
- the distance between the antenna elements strongly impacts on the channel correlation: the more distance increases, the more uncorrelated the antenna elements become;
- as the correlation between the antennas increases, a consequent improvement of beamforming performance is progressively achieved. On the other hand, increasing the correlation between antennas, the spatial multiplexing gain is reduced; one approach to improve spatial multiplexing performance can consist in transmitting the signal only on a subset of the antenna elements.

4.5 A ChannelNet model for New Radio

The NR link simulator presented in the previous section is the instrument we used to explore new DL applications in 5G scenarios. Specifically, in this section we introduce a channel estimation solution inspired by the ChannelNet model presented in `Section 4.3`. Our approach, named **NR-ChannelNet**, maintains substantially the same structure of the CNN illustrated in paper [21], but it introduces some variations in the training procedure and it attempts at generalizing the model to multiple use cases:

- first of all, we try to make the model applicable also to MIMO scenarios, differently from the work proposed in reference [21] which focuses only on the channel matrix along a single transmitter/receiver antenna pair;
- in addition, we try to enrich our training datasets by performing link simulations at different levels of SINR: instead of fixing two values of SINR for the training process, we change it with a finer granularity, with the purpose of obtaining models with better performances also in scenarios where the quality of the channel significantly oscillates;

- then, we further keep distance from the training approach described in reference [21], analyzing an alternative method that differs in the way real and imaginary parts are processed: the idea comes from another paper, [23], in which the authors suggest a double-channel scheme. Finally, the results obtained with this alternative approach, illustrated in the following pages, are compared with the ones obtained by applying the original ChannelNet.

We build and train the neural network models in `Python` using `Keras`, a `TensorFlow`'s open-source high-level API. Once completed the training phase, we developed a deep learning-based channel estimation block inside a Telecom Italia proprietary software link simulator, implemented in `MATLAB`®. This block basically substitutes the pre-existing low-rank 2D-MMSE estimator: it imports a pre-trained neural network model in `MATLAB` environment and predicts the channel estimate exploiting the functionalities of the `MATLAB` Deep Learning tool.

4.5.1 Training process

`Table 4.4` reports the configuration of the simulator parameters that are of particular interest in our experiments.

It is worth noting that the system bandwidth selected for our experiments, 1.4 MHz, is not provided by the 5G standard, but only by LTE: behind this choice there are reasons of computational complexity, since a larger band would lead to a significant increment in simulation time.

As regards the transmitter and receiver antenna systems, we consider two different configurations, shown in `Table 4.5`. N_{TX} and N_{RX} represent the number of transmitter and receiver antenna respectively, while N_1 and N_2 stand for the number of antenna elements on the azimuth and elevation plane; N_{pol} instead indicates the number of polarizations for each antenna element.

The training dataset collected for our NR-ChannelNet model consists of $\{input, output\}$ pairs:

- each input entry is a matrix containing the channel response values that are estimated in correspondence of DM-RS symbols ($\boldsymbol{p_{DMRS}}$);
- the correspondent desired output is the beamformed-version of the perfect channel matrix that is generated by the CDL-B model ($\boldsymbol{HW_{id}}$).

It is worth observing that our DL-based approach can be classified as supervised, since the desired output of the model is known for each input value.

Table 4.4: Simulation parameters for NR-ChannelNet training

Simulation parameters	
Transmission direction	Downlink
Carrier frequency	3.64 GHz
System bandwidth	1.4 MHz
Subcarrier Spacing	15 kHz
TTI duration	1 ms
iMCS	21
(Modulation, Coding rate)	(64-QAM, 0.694)
TBS	4736 bit
Number of FFT samples (NFFT)	128
Number of SUBcarriers used (NSUB)	72
Number of PRB (NPRB)	6
Cyclic Prefix (CP) Type	Normal
Number of OFDM symbols per TTI (NOFDM)	14
Number of transmission layers	1
Number of codewords	1
DM-RS Type	Type 1
DM-RS Mapping Type	Type A
Number of DM-RS front-load symbols	1
Number of additional DM-RS	0
Channel Model	CDL-B
Delay Spread model	Nominal

Table 4.5: Antenna systems configurations considered for NR-ChannelNet training

Antenna system configuration					
Configuration	N_{TX}	N_{RX}	N_1	N_2	N_{pol}
8×1	8	1	2	2	2
32×2	32	2	4	4	2

Selecting the DM-RS configuration type 1 and considering one single transmission layer per codeword, the size of the matrix $\boldsymbol{p_{DMRS}}$ is $N_{RX} \times NOFDM_{DMRS}NSUB/2$, where NSUB corresponds to the number of subcarriers used for data transmission

and $NOFDM_{DMRS}$ is the number of OFDM symbols used for DM-RS allocation. Remembering the allocation scheme illustrated in Figure 4.3, if no additional OFDM symbol is used, only the Resource Elements (REs) on the odd subcarriers in correspondence of the third OFDM symbol contains DM-RS pilots.

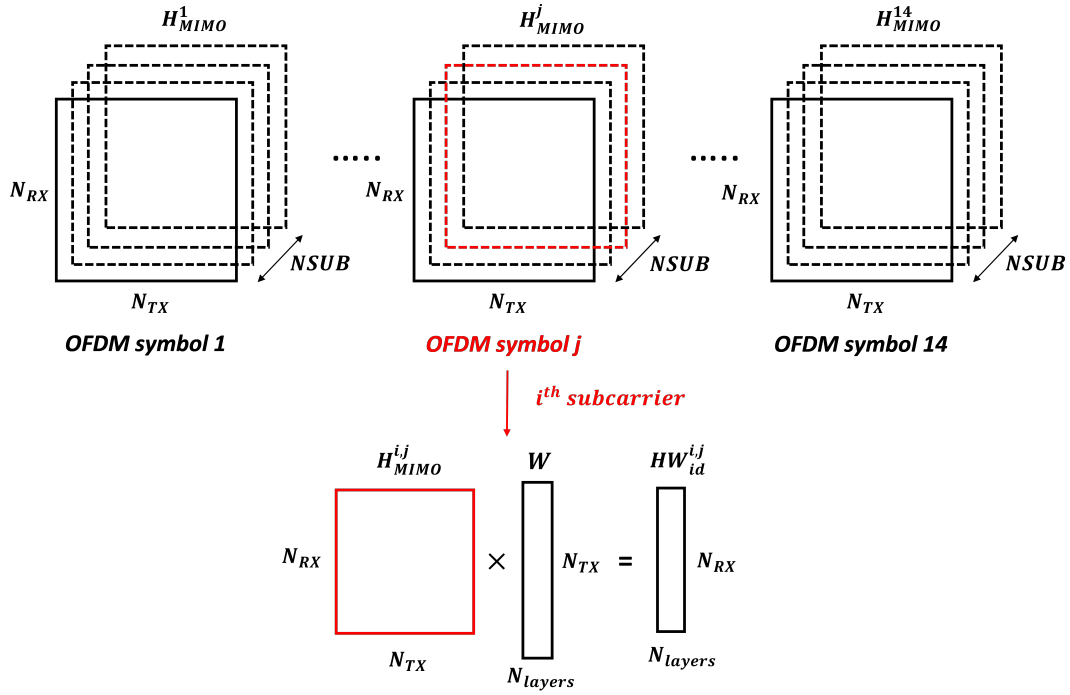


Figure 4.11: HW_{id} matrix computation

The matrix HW_{id} should not be confused with the channel time-frequency response image $\boldsymbol{H}$, introduced in Section 4.3: the matrix $\boldsymbol{H}$, in fact, contains the channel attenuation values for a SISO link and therefore it has size $NSUB \times NOFDM$; our matrix HW_{id}, instead, represents the beamformed channel matrix, of size $N_{RX} \times N_{layers} \times NSUB \times NOFDM$, but considering a single transmission layer, the size reduces to $N_{RX} \times NSUB \times NOFDM$. For each subcarrier i and OFDM symbol j, $\boldsymbol{HW}^{i,j}_{id}$ is computed as the multiplication of the CDL-B channel matrix $\boldsymbol{H}^{i,j}_{MIMO}$, of size $N_{RX} \times N_{TX}$, by the beamforming vector $\boldsymbol{W}$, of size $N_{TX} \times N_{layer}$, as Figure 4.11 shows. As a consequence, in our model also a third dimension must be taken into account: in fact, while in paper [21] only single receiver antenna scenarios are considered, our model is aimed to work also with configurations where $N_{RX} > 1$. To deals with this challenge, we adopt two different approaches:

1. single-channel approach: considering channel images associated to different receiver antennas as distinct data point on the same input channel: with this

approach only 2D gray-scale images are considered as input of the CNN;

2. multi-channel approach: making our NR-ChannelNet model able to deal with colored input images: in few words, each different receiver antenna constitutes a different color channel.

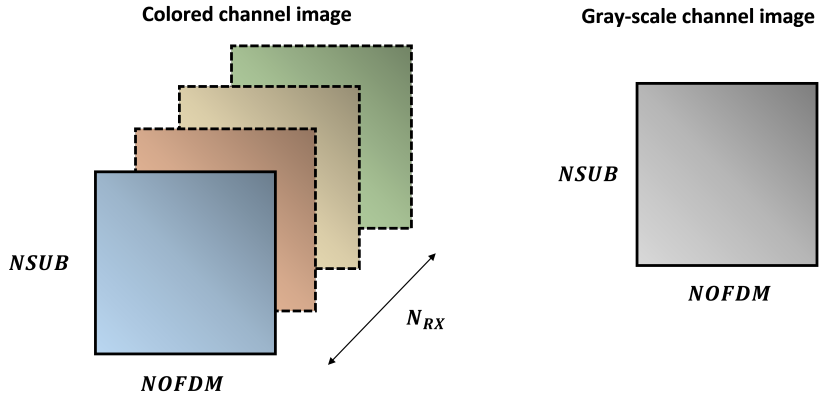


Figure 4.12: Colored and gray-scale channel images

`Figure 4.12` helps to clarify this concept. This two approaches will be compared in terms of performance in the following section, but an important difference between the two can be highlighted right away: with the multi-channel approach, M simulation cycles provide M data points for the training set, while adopting the single-channel method the training instances obtained are MN_{RX}. This means that, fixed the size of the training dataset, the simulation time required with the first approach would be reduced by a factor N_{RX}.

In conclusion, there exist fundamental differences with respect to the work presented in paper [21] that make the NR-ChannelNet proposed in our thesis project applicable to a wider range of cases, including MIMO communications.

Input pre-processing: interpolation and scaling

Before passing through the first layer of the NR-ChannelNet model, all the matrices p_{DMRS} contained in the dataset are subject to a two-step preprocessing that consists of:

1. **interpolation**;

2. **min-max normalization**.

First, we place the values of p_{DMRS} on a time-frequency grid and we apply an interpolation algorithm on the real and imaginary part separately, to pre-compute the

channel attenuation in positions where DM-RS pilots are not present; in particular, we adopt the Radial Basis Function (RBF) interpolation implemented in `Python scipy` library. The figures below illustrate the interpolation process: `Figure 4.13` shows the channel attenuation values estimated in correspondence of the DM-RS pilots, while `Figure 4.14` displays the complete interpolated channel matrix. We can denote the matrix resulting from this interpolation step as $\boldsymbol{HW_{DMRS}}$. The presence of the matrix $\boldsymbol{W}$ in the name reminds that, when the DM-RS pilots are compensated at the receiver side, the effect of the beamforming matrix $\boldsymbol{W}$ applied at the transmitter side is not eliminated: this means that, when we interpolate the DM-RS values, what we get is an estimate of the beamformed channel matrix.

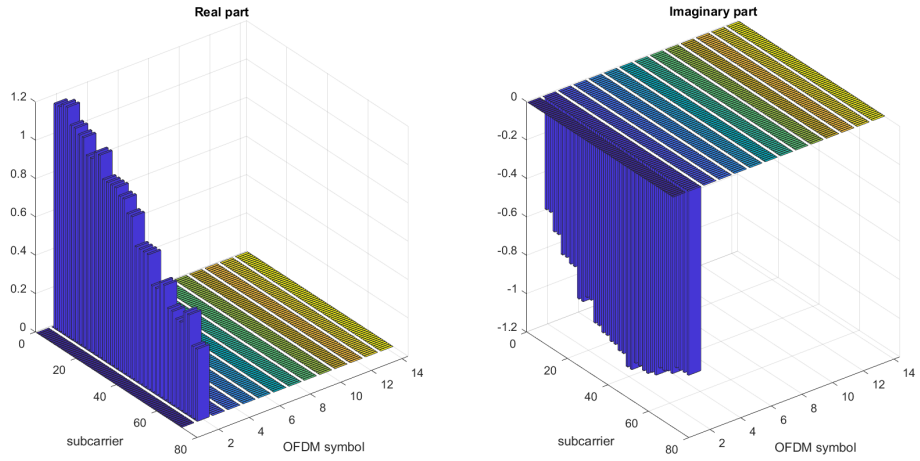


Figure 4.13: Compensated DM-RS pilots on time-frequency grid at SINR = 20 dB

Once we obtain the matrix $\boldsymbol{HW_{DMRS}}$, we rescale the values of both its real and its imaginary parts in the range [0,1] (`Figure 4.15`). This is a crucial step in the preprocessing pipeline, since without data normalization, the objective function of the DL algorithm does not work properly in most cases. Data normalization or feature scaling is a technique that normalizes the range of values characterizing independent variables or features [W5]. As book [19] explains, when features are measured on different scales the optimization algorithm will be governed by the feature with the broadest range, since the weights will be mostly optimized basing on its errors. To counteract this issue, we perform a min-max normalization, according to the following expression:

$$\boldsymbol{x_{norm}} = \frac{\boldsymbol{x} - min(\boldsymbol{x})}{max(\boldsymbol{x}) - min(\boldsymbol{x})}. \tag{4.22}$$

The resulting matrix is denoted as $\boldsymbol{H_{DMRS,scal}}$.

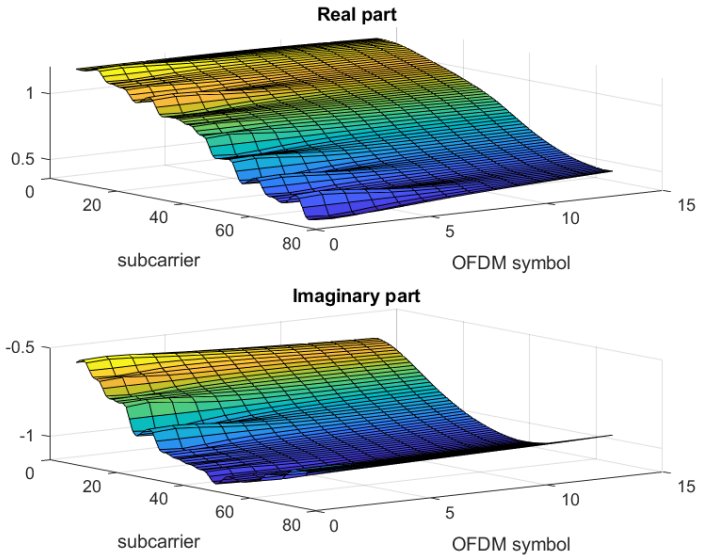


Figure 4.14: Interpolated time-frequency grid at SINR = 20 dB

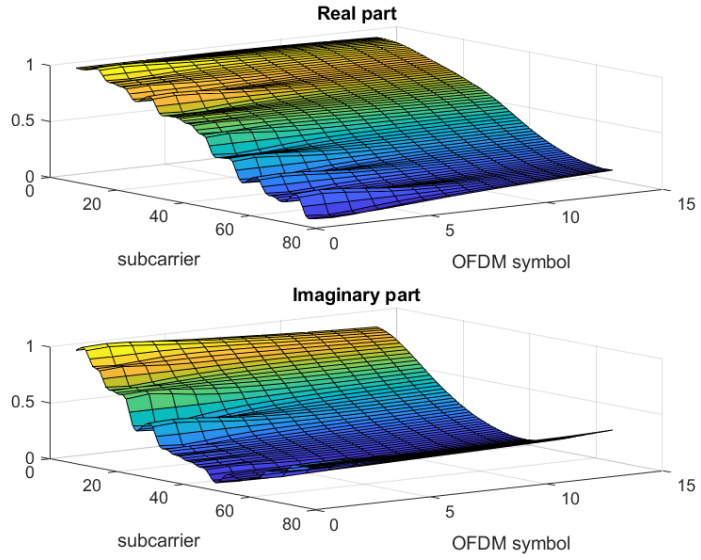


Figure 4.15: Rescaled time-frequency grid at SINR = 20 dB

In parallel, also the desired output matrices $\boldsymbol{HW}_{id}$ must be subject to the same scaling operation (`Figure 4.16`); obviously, no prior interpolation is performed since the output matrices are entirely derived from the channel model.

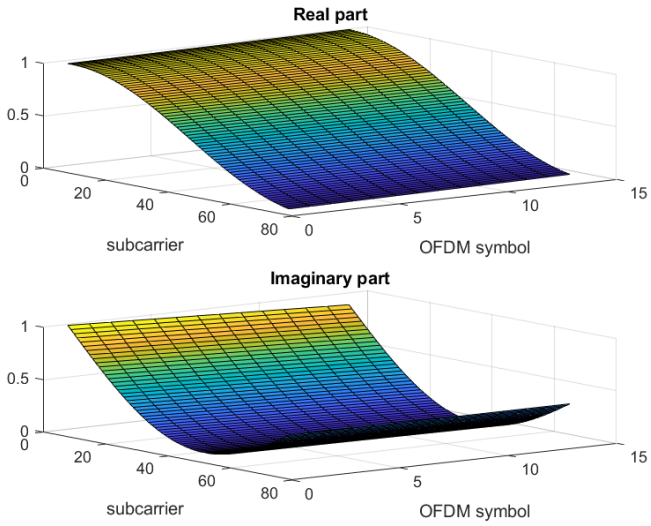


Figure 4.16: Rescaled perfect channel matrix

Real and imaginary parts

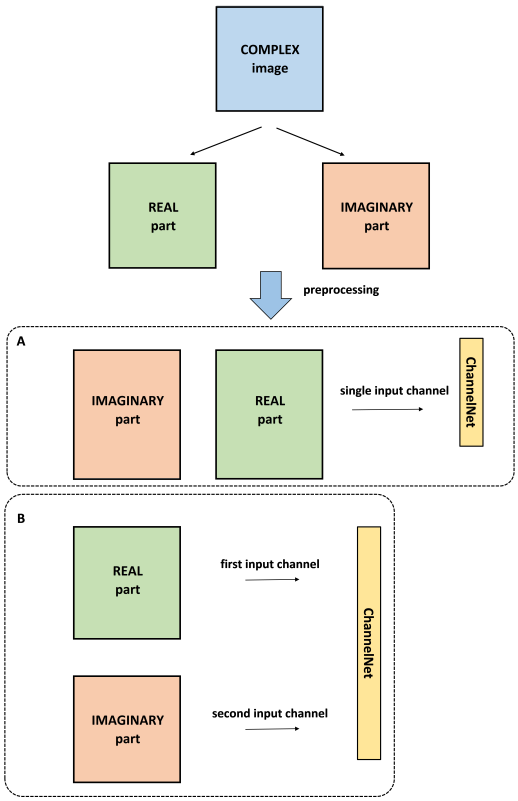


Figure 4.17: Real and imaginary parts as single or double input channel

The authors of paper [21] consider the real and the imaginary parts of the input matrix $\boldsymbol{H_{DMRS}}$ and of the desired output matrix $\boldsymbol{HW_{id}}$ as distinct entries of the training dataset: this means that, by simulating N TTI, $2N$ pairs $(\boldsymbol{H_{DMRS}}, \boldsymbol{HW_{id}})$ are obtained. Differently from this approach, in paper [23] the real and the imaginary parts are considered as distinct colors of the same image, conveyed through the network on separate channels: to realize this, the authors double both the third dimension of the input layer and the number of feature maps at the output of the last convolutional layer. `Figure 4.17` summarizes the method proposed in paper [21] (A) and the one suggested in paper [23] (B). Even if the approach proposed in reference [23] does not concern the channel estimation framework, we exploit it to develop a second NR-ChannelNet version, named 2-channel, with the purpose of providing a valid meter of comparison to evaluate the performance of the original ChannelNet algorithm.

Variability of channel quality

The $\boldsymbol{H_{DMRS}}$ matrices are collected by simulating different conditions of the channel: in particular, we modify the SINR value in order to test the behavior of our DL-based model at different levels of channel quality.

The authors of paper [21] limit themselves to train the ChannelNet for two different SINR values:

- for low SNR values, the network is trained at 12 dB of SINR. This trained model takes the name of *deep-low SNR*;
- for higher SNR values, the network is trained at 22 dB of SINR. This trained model takes the name of *deep-high SNR* .

Substantially, they divide the SINR range into two regions, fixing an intermediate threshold: below this threshold, the low-deep SINR model is employed; above it, the high-deep SINR model is chosen. According to the results reported in the paper, the authors obtain acceptable performance even with this simplified approach. Anyway, several unsolved issues persist: first of all, if simulations were extended also to SINR values above 20 dB, other models would have to be trained; furthermore, adopting a different wireless channel model, maybe more complex than VehA and SUI5 models considered in reference [21], a finer SINR granularity might be required in the training process. In the light of this, we prefer to try two different approaches:

- on one hand, we select a set of different SINR values, {0, 10, 20, 50 dB}, and we train a different NR-ChannelNet model for each of these values;

- in parallel, we define two SINR regions, [0;20] and [20;50] dB. For each region, we collect our training data varying the SINR inside the corresponding range of values: for the first SINR region, the SINR values considered for the simulations are 0 dB, 10 dB and 20 dB; for the second one instead, training data are collected at SINR 20 dB and 50 dB. With this second approach, we attempt at improving the performance of our NR-ChannelNet in scenarios where the SINR oscillates significantly: by increasing the variability of the training data, we try to obtain a more generalizable model, more insensitive to eventual channel variations.

The simulation results obtained by employing these two alternative approaches will be analyzed in the following section.

4.6 Simulation results

The performance of the NR-ChannelNet model, in all the variants illustrated, are evaluated after simulation campaigns conducted with the NR link level simulator presented in Section 4.4. Table 4.4 reports the main parameters of interest concerning the simulator settings, while Table 4.6 and Table 4.7 show the hyperparameters adopted for the training and testing process of our model.

It is important to avoid misunderstanding about the notation used in the following pages, therefore some clarifications are required. When 8×1 antenna configuration is assumed, two different NR-ChannelNet models are considered:

- 1-channel: the label refers to the fact that, in this model, real and imaginary parts of the channel images are fed to the network through a single input channel;

- 2-channel: real and imaginary parts of the channel images are conveyed through the network on two distinct channels.

Adopting, instead, a 32×2 antenna configuration, the model variants considered are:

- 2-channel: real and imaginary parts of all the channel images, independently on the receiver antenna they are relative to, are conveyed on the same two input channels;

- multi-channel: real and imaginary parts of the channel images relative to different receiver antenna are conveyed to the network on $2N_{RX}$ distinct channels.

Table 4.6: Simulation parameters for 8x1 configuration

Parameter	1-channel model	2-channel model
Training set	500 sim of 100 TTI	1000 sim of 100 TTI
Validation set	50 sim of 100 TTI	100 sim of 100 TTI
Testing set	100 sim of 100 TTI	100 sim of 100 TTI
Learning rate	0.001	0.001
Loss function	MSE	MSE
Optimizer	Adam	Adam
SRCNN epochs	300	400
DnCNN epochs	200	200
DnCNN depth (D)	4	4

Table 4.7: Simulation parameters for 32x2 configuration

Parameter	2-channel model	multi-channel model
Training set	1000 sim of 100 TTI	2000 sim of 100 TTI
Validation set	100 sim of 100 TTI	200 sim of 100 TTI
Testing set	100 sim of 100 TTI	100 sim of 100 TTI
Learning rate	0.001	0.001
Loss function	MSE	MSE
Optimizer	Adam	Adam
SRCNN epochs	300	400
DnCNN epochs	200	200
DnCNN depth (D)	4	4

Some of the plots shown in the following pages report the Normalized MSE (NMSE) between the perfect beamformed channel matrices and the matrices estimated by means of our pre-trained NR-ChannelNet models; the NMSE is computed as

$$NMSE = \frac{||\boldsymbol{HW_{id}} - \boldsymbol{HW_{DMRS}}||_2^2}{||\boldsymbol{HW_{id}}||_2^2} \tag{4.23}$$

where $\boldsymbol{HW_{DMRS}}$ and $\boldsymbol{HW_{id}}$ are the estimated and the perfect channel images respectively, as explained in `Subsection 4.5.1`. The values reported represent the values mediated first on the different TTIs and then on the different testing simulations.

The tables attached in `Appendix A` report some statistics again mediated both in time and on the different simulations performed:

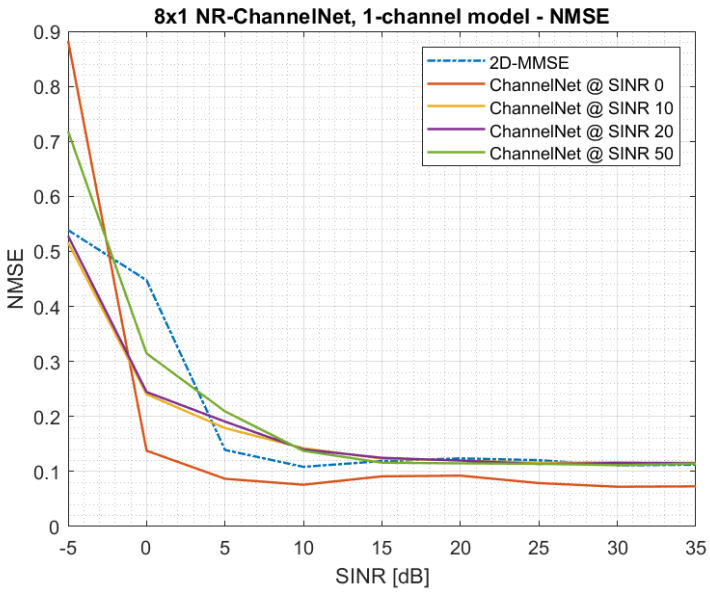
- σ_{NMSE}: this value represents the standard deviation of the mean NMSE values obtained in the various simulations. The standard deviation can be considered a useful indicator, since it measures how variable the algorithm performances are:

large values of standard deviation would suggest a certain degree of unreliability in the results shown;

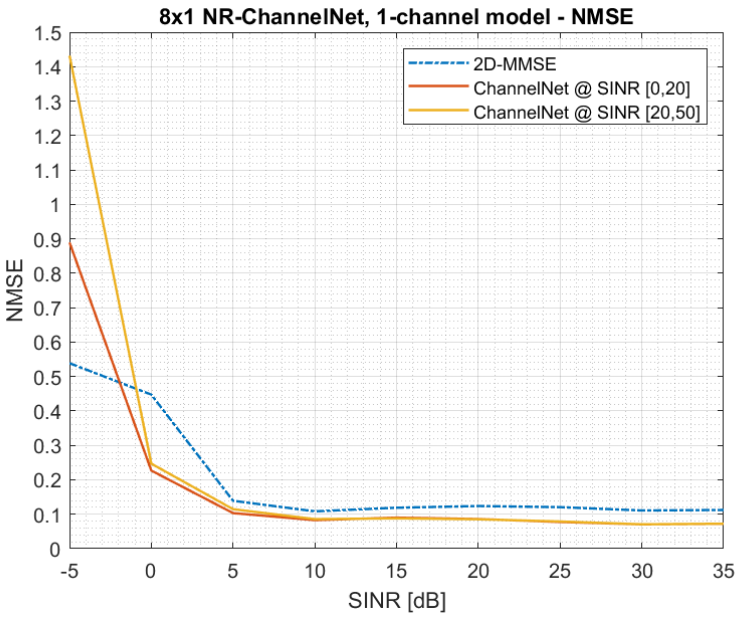
- `Thr`: throughput expressed in [Mbit/s];
- `Dec BER`: Bit Error Rate at the end of the decoding process;
- `BLER`i: BLock Error Rate (BLER) at the i-th transmission attempt. For our simulation, the NR link simulator is configured with a maximum number Hybrid-Automatic Repeat reQuest (H-ARQ) transmissions equal to 4 and this is the reason why only `BLER1`, `BLER2`, `BLER3` and `BLER4` are considered.

4.6.1 MISO scenario: 8 X 1 antenna configuration

The plots below refer to the simulation results obtained in case of 8×1 antenna configuration: in particular, `Figure 4.18` and `Figure 4.19` show the trend of NMSE values for the 1-channel and the 2-channel NR-ChannelNet variant respectively. The graphs in `Figure 4.18a` and `Figure 4.19a` report the results of the models trained with data collected at constant SINR levels; the NMSE curves shown in `Figure 4.18b` and `Figure 4.19b`, instead, are relative to NR-CsiNet variants trained at mixed SINR levels, according to the second training approach described in `Subsection 4.5.1`.

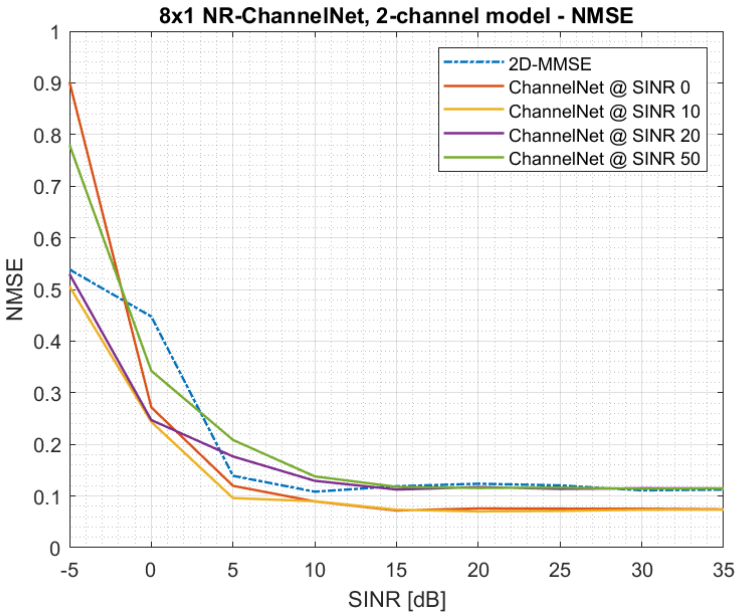


(a) NR-ChannelNet 1-channel single SINR values

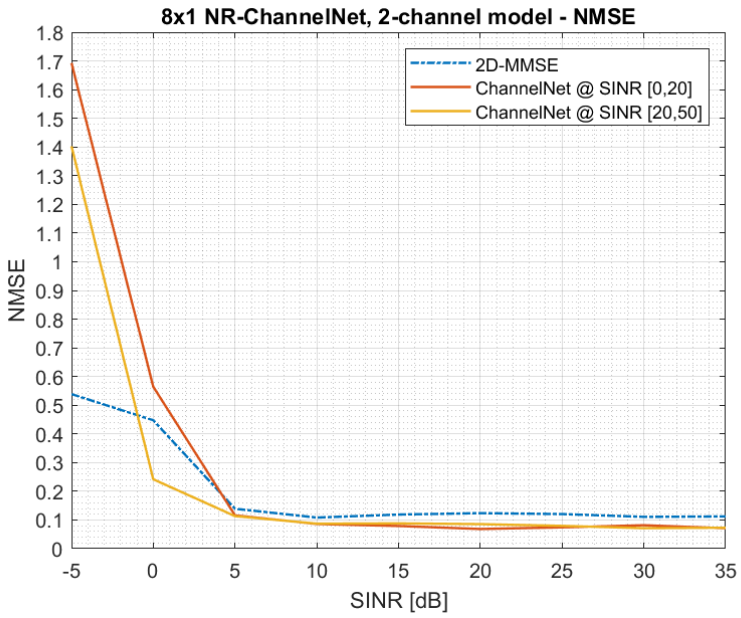


(b) NR-ChannelNet 1-channel mixed SINR values

Figure 4.18: NR-ChannelNet 1-channel model NMSE - 8x1 configuration



(a) NR-ChannelNet 2-channel single SINR values



(b) NR-ChannelNet 2-channel mixed SINR values

Figure 4.19: NR-ChannelNet 2-channel model NMSE - 8x1 configuration

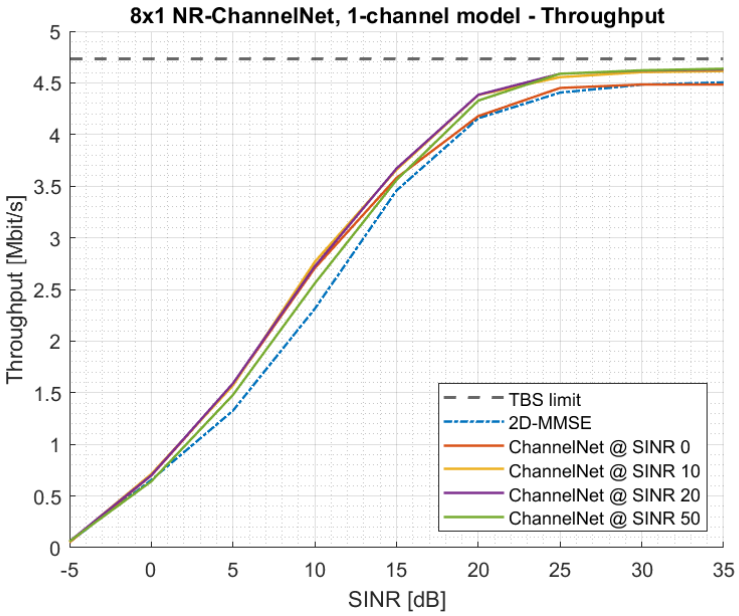
It is possible to note that there are no significant performance disparities between the various models presented: maybe, the 2-channel approach would seem to be slightly more performing and, at low SINR levels, the mixed-SINR approach would appear to be less convenient.

An interesting observation concerns the NMSE standard deviation. Looking at the data reported in `Section A.1`, σ_{NMSE} values increase with the decrease of the channel SINR level, as expected: when the noise component is predominant with respect to the useful signal, performance trend acquires a random behavior, which translates into higher standard deviation values.

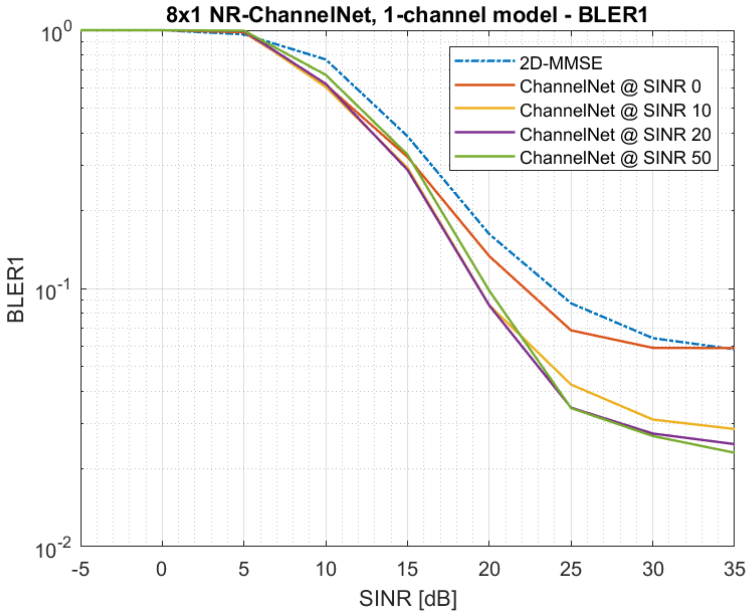
The plots displayed in the following pages are useful to summarize and compare the results obtained, showing the curves of throughput and BLER1 for all the different trained models. It is worth noting that the graphs relative to the throughput report also an horizontal straight line, representing the maximum throughput level that can be achieved given the selected Transport Block Size (TBS); the distance of the throughput curve from the threshold given by the TBS value can be a useful performance indicator. As a matter of fact, a comment must be made: as it can be observed in the plots, the throughput curves never reach the maximum possible value, even at very high SINR levels. The reason lies in the TBS computation: the NR link simulator adopted for our experiments calculates the TBS value only once, at the beginning of the simulation. The formula used for the computation does not take into account that, at each CSI

reporting period, some resource elements in the time-frequency grids must be reserved for Channel State Information-Reference Signal (CSI-RS) pilots and cannot be allocated for data. As a result, on TTIs where CSI-RS are transmitted, the available resource elements are not sufficient to allocate both the computed TBS and CSI-RS pilots: anyway, to fit the encoded transport block of data in the available resource elements, some redundant bits are sacrificed, with a consequent increase of the code rate. A lower degree of protection could justify a positive decoded BER even in high channel quality conditions.

Analyzing the plots below, it is evident that the SINR level at which the considered model as been trained has negligible impact on performances; the only exception would seem to be the NR-ChannelNet trained at 0 dB of SINR, which turns out to be the worst performing. As a consequence, the idea of mixing data collected at different SINR levels, with the purpose of training models more insensitive to the channel quality oscillation, does not prove particularly useful. Moreover, 1-channel and 2-channel approaches turn out to achieve very similar results, comparable also with those obtained by the standard 2D-MMSE algorithm.

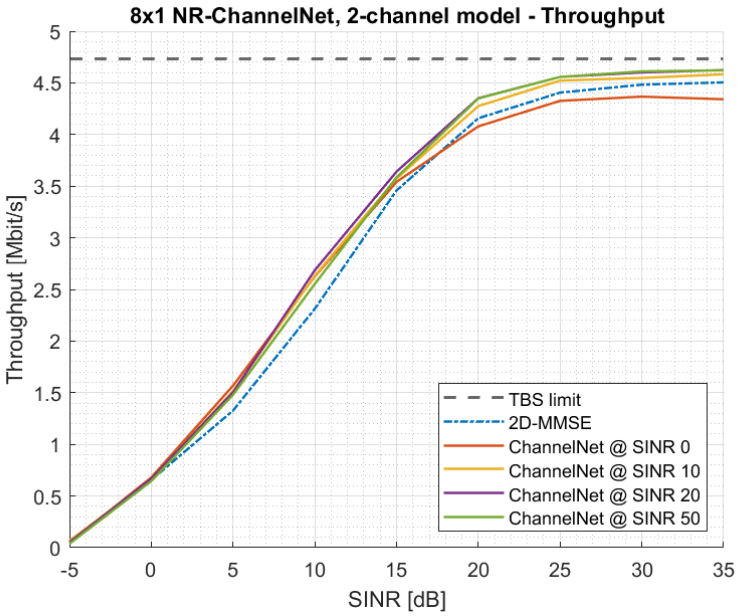


(a) Throughput

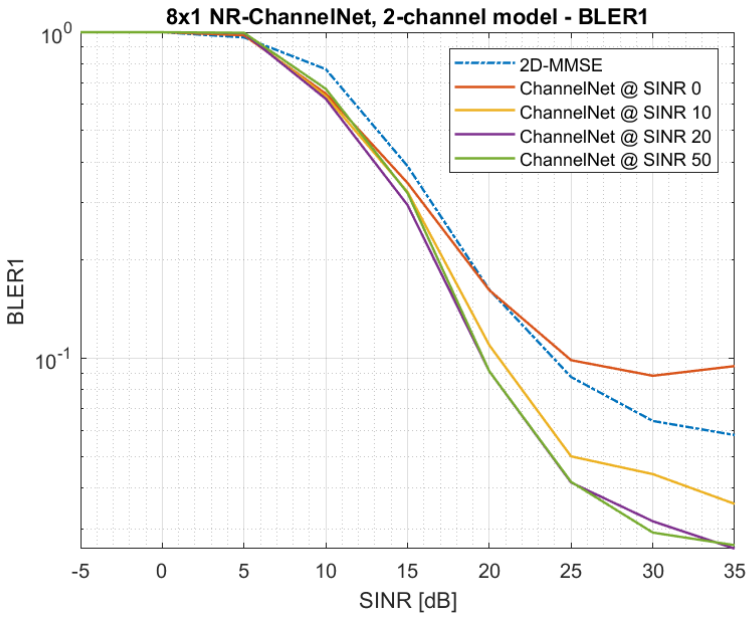


(b) BLER1

Figure 4.20: NR-ChannelNet 1-channel model - 8x1 configuration

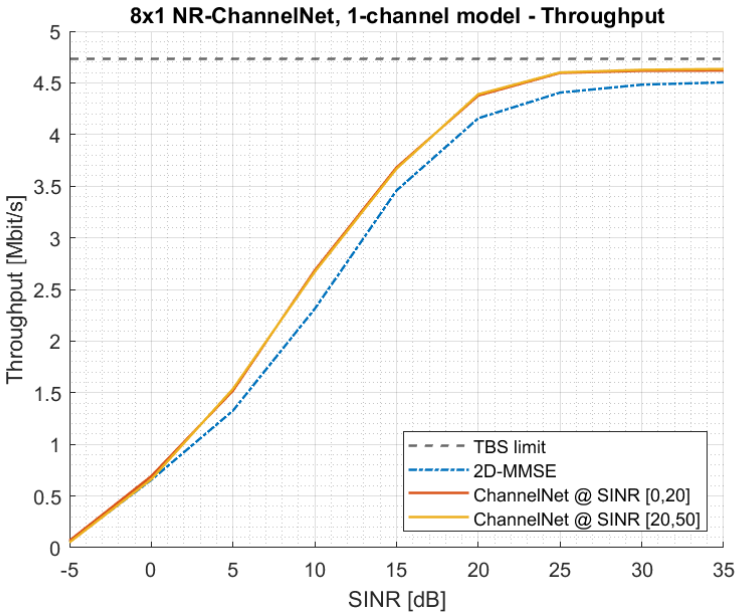


(a) Throughput

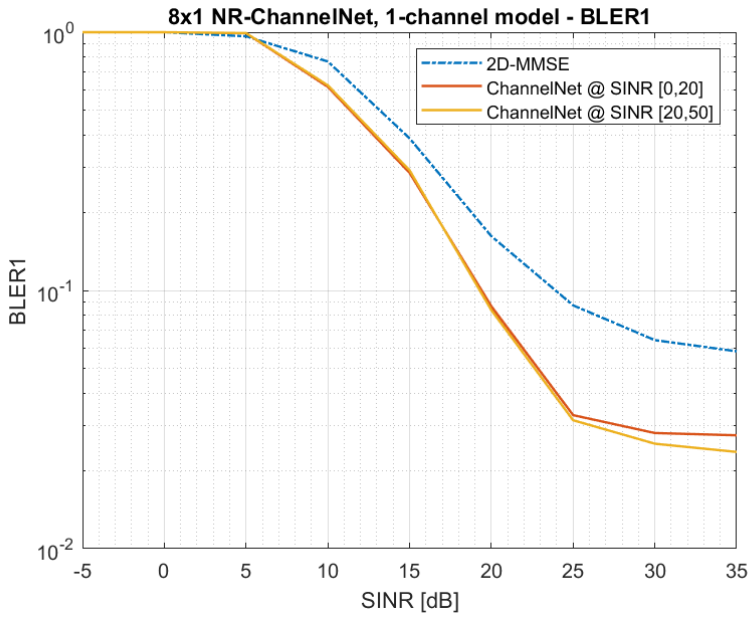


(b) BLER1

Figure 4.21: NR-ChannelNet 2-channel model - 8x1 configuration

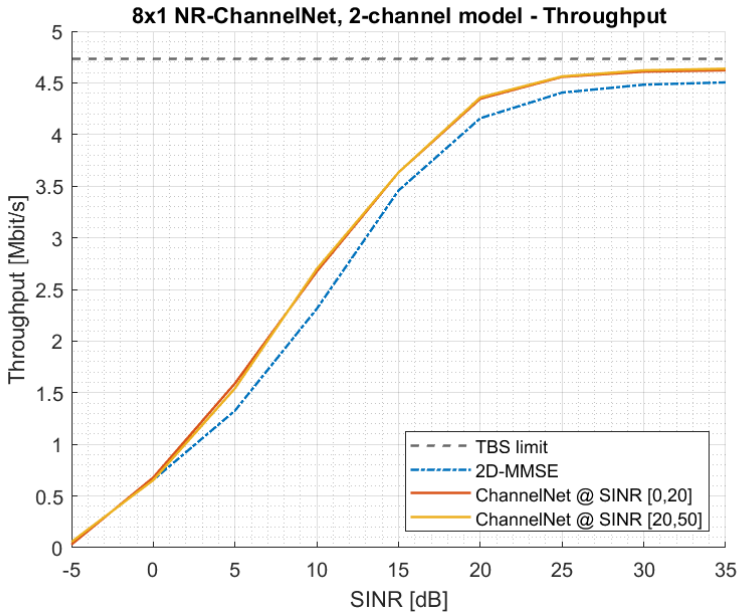


(a) Throughput

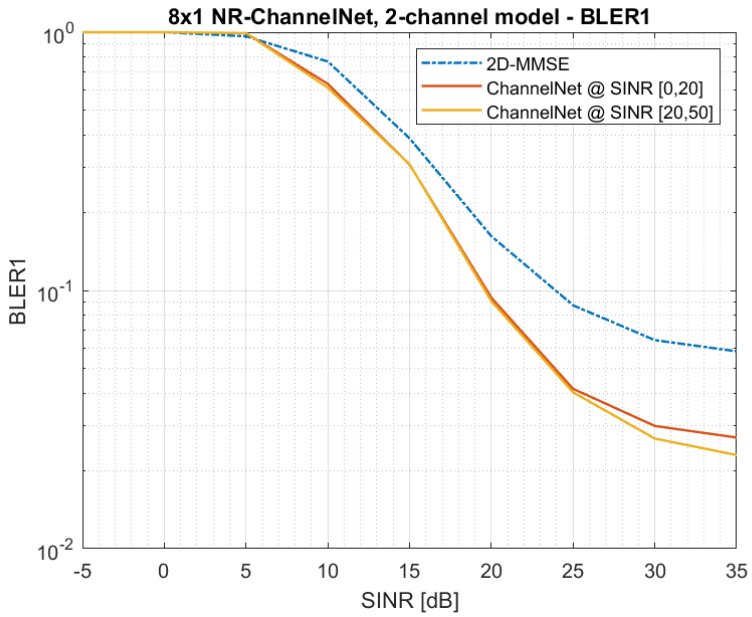


(b) BLER1

Figure 4.22: NR-ChannelNet 1-channel model, mixed SINR values - 8x1 configuration



(a) Throughput



(b) BLER1

Figure 4.23: NR-ChannelNet 2-channel model, mixed SINR values - 8x1 configuration

To conclude, the figures above demonstrate that our deep learning-based approach is absolutely competitive with the low-rank 2D-MMSE algorithm; indeed, some variants of NR-ChannelNet achieve slightly better results.

4.6.2 MIMO scenario: 32 X 2 antenna configuration

This subsection refers to simulation experiments performed with a 32×2 antenna configuration. With this second configuration, characterized by an increased number of antennas both at the transmitter and at the receiver side, we certainly expect better results: a larger number of transmitter and/or receiver antennas can provide additional diversity against fading on the radio channel [W25]; not only this, but a greater availability of antennas results in a grater beamforming gain, due to an improved ability to orient the beam in the direction of the receiver. In particular, by multiplying the parameter N_{TX} by a factor of 4, the beamforming gain achieved for the transmitted signal increases proportionally, by 6 dB.

That said, it is possible to find an effective match in the graphs shown on the following pages.

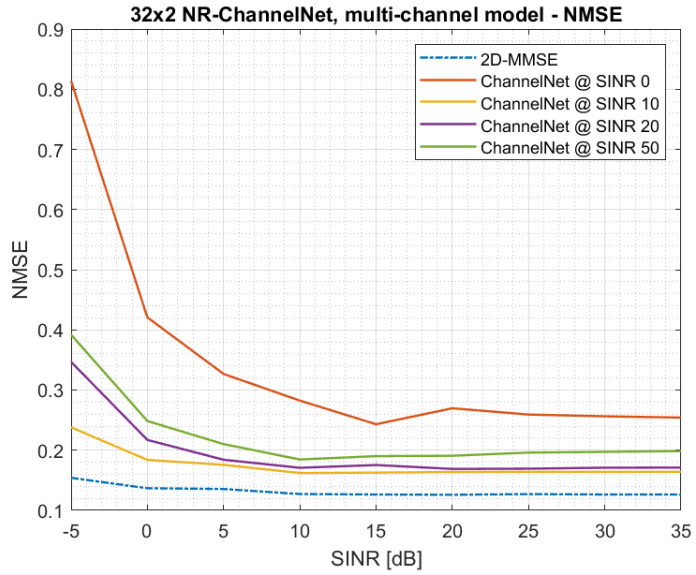


Figure 4.24: NR-ChannelNet multi-channel model NMSE - 32x2 configuration

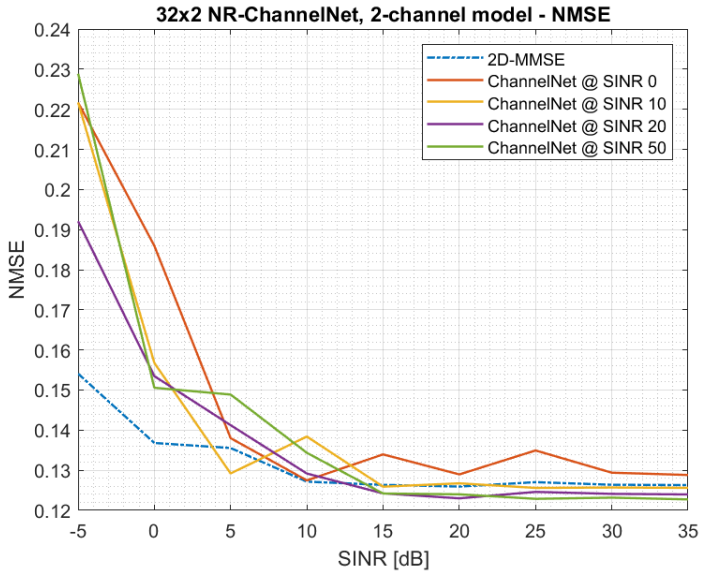
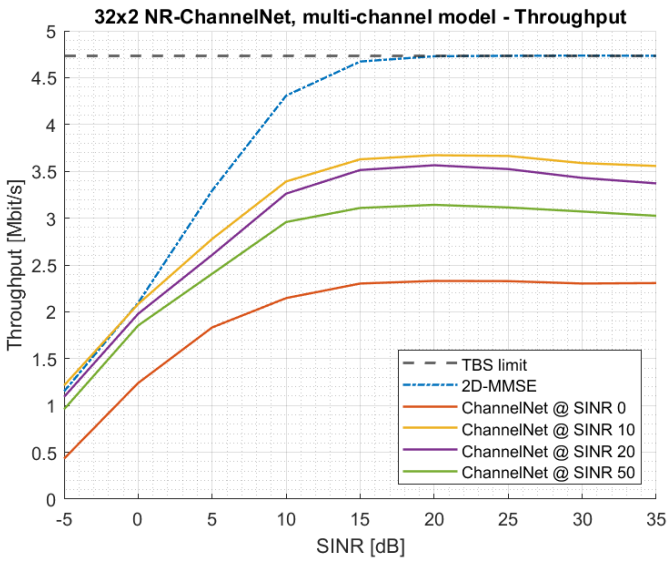


Figure 4.25: NR-ChannelNet 2-channel model NMSE - 32x2 configuration

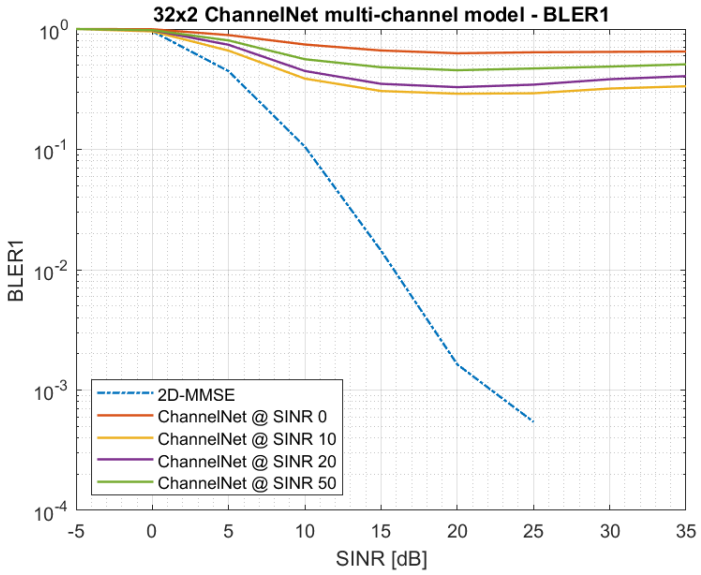
What's immediately striking about these plots is the fact that the standard 2D-MMSE algorithm achieves better performances in terms of NMSE. However, this apparent improvement is not reflected in the data reported in `Section A.2`: in fact, apart from some exceptions, the decoded BER and the BLER values demonstrate equivalent and, in some cases, even slightly better performances by the NR-ChannelNet models. From

this fact an important observation can be drawn: the NMSE is not a totally reliable performance indicator. Instead, the set of parameters reported in the tables attached in the Appendix A.2 can give us real and trustable indications to compare the different models.

As for the 8×1 configuration, the following pages report some useful graphs to summarize and compare the results obtained: throughput and BLER1 extracted form the tables in Section A.2 can help to evaluate the different NR-ChannelNet variants considered. Both the graphs in Figure 4.26 and the results stored in Table A.15 clearly show that the NR-ChannelNet multi-channel model trained at $SINR = 0$ dB performs very poorly with respect to the correspondent 2-channel variant; this is also confirmed by the NMSE plot in Figure 4.24, where the line corresponding to the model assumes rather high values. This fact is probably due to overfitting occurrence we encountered in the training phase. Our guess is that, when the input channel images are particularly noisy, the training set collected and fed to the convolutional network essentially contains only noise; as a consequence, larger datasets are needed to finalize the learning process. Moreover, keeping in mind that with the multi-channel approach the number of input channels grows linearly with the number of receiver antennas, it is clear that also the number of learnable parameters increases proportionally. As a consequence, our dataset may not be larger enough to avoid overfitting phenomena. The reason why this problem does not occur for the other multi-channel models could lie in the fact that also the quality of the dataset plays an important role: channel matrices collected at higher SINR levels contain more useful information, making a smaller amount of training data sufficient to accomplish the learning process.

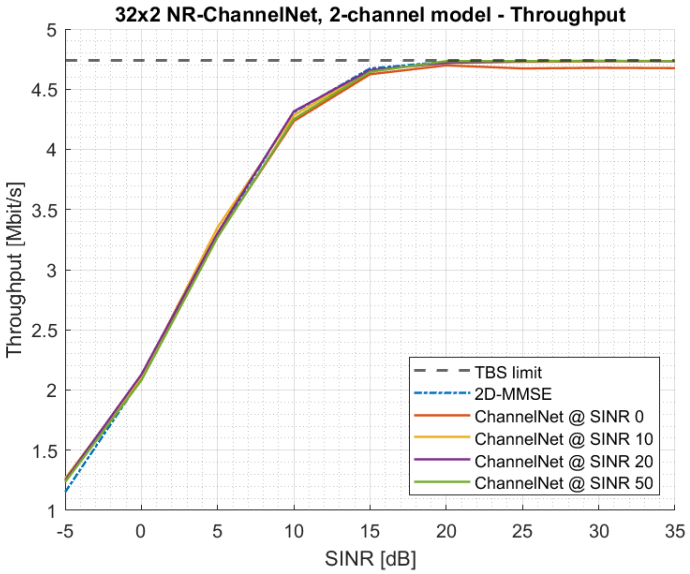


(a) Throughput

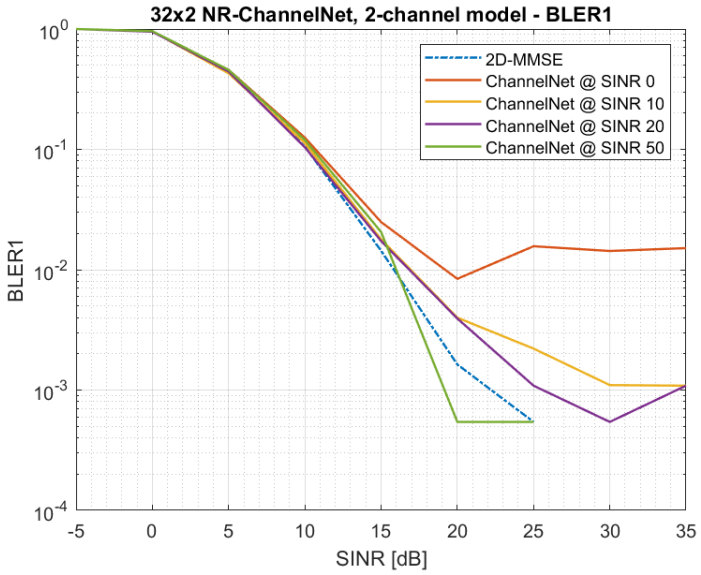


(b) BLER1

Figure 4.26: NR-ChannelNet multi-channel model - 32x2 configuration



(a) Throughput



(b) BLER1

Figure 4.27: NR-ChannelNet 2-channel model - 8x1 configuration

Anyway, we can state that, in general, 2-channel models achieve better performances. But this is not the only reason why this kind of approach should be preferred. For our experiments, we have considered a maximum of two receiver antennas for complexity reasons; however, in Massive MIMO scenarios, N_{RX} can significantly increase and, considering the multi-channel approach, this would lead to a proportional increase in the number of both input/output channels and learnable parameters of our convolutional neural networks. The 2-channel approach is certainly more scalable with the number of receiver antennas and this is a desirable characteristic in a 5G context.

Comparing the results obtained with the two different antenna configurations considered, it is quite evident that NR-ChannelNet models bring more significant improvements for the 8×1 scenario: this sounds reasonable since, when a single receiver antenna is available, system performances are more sensitive to the channel estimation accuracy and thus there is more room for improvement.

Chapter 5

CSI feedback reporting for MIMO scenarios: a Deep Learning approach

As reference [5] explains, the availability of information about the colorful characteristics of the communication channel has a fundamental impact on many transmission features in most of the modern radio-access technologies. This framework of information can consist in a rough estimate of the radio link path loss, that is useful for example to adjust the transmitted power, to a very precise knowledge of the channel amplitude and phase [5]. Measurements and estimates can be performed either by the transmitter or by the receiver side: focusing on the downlink case, the knowledge of the channel characteristics can be acquired by means of device measurements or by the network itself, depending on the particular scenario:

- when working in FDD mode, downlink and uplink contexts could potentially be significantly different. As a consequence, to obtain trustable information about the downlink channel, the measurements must be necessarily acquired by the user device and then reported to the network; the network will exploit this feedback to properly set some transmission parameters for future downlink transmissions;
- in case of TDD communications, when downlink and uplink transmissions experience the same channel characteristics, a feedback from the device is not necessary, since the network itself can obtain useful information about the downlink features of interest by measuring them in the uplink direction.

In general, the evaluation of the radio environment takes the name of **channel sounding** and it requires the adoption of specific **reference signals (RS)** on which the

receiver can perform measurements.

Many sources claim that, with the introduction of Massive MIMO technologies in 5G systems, the potential gain from an accurate estimation of channel characteristics has considerably increased and channel sounding has become a critical aspect. Moreover, as the authors of paper [23] make notice, the most used approaches to reduce the feedback overhead, based on codebooks or vector quantization, elaborate feedback quantities whose dimensions scale with the number of antennas, making these techniques unsuitable in Massive MIMO scenarios. These criticisms have propelled new studies aiming at overcoming the traditional approaches with Deep Learning-based solutions.

The first sections of this fifth chapter provide a close examination of 5G **Channel State Information reference signals** (**CSI-RS**), which are used by terminals to estimate the Channel State Information relative to the downlink channel. In the following sections, an overview on information reporting techniques is presented, to illustrate the limits of some traditional approaches. Finally, a new Deep Learning-based technique is proposed for Channel State Information feedback reporting: starting from the model proposed by the authors of paper [23], a 5G-compliant CsiNet is presented, accompanied by some simulation results to evaluate its performance in some specific scenarios.

5.1 Channel State Information reference signals

One of the key design principles of NR is to avoid, as far as possible, "always-on" signals [5]. In release 8 of the LTE standard, channel sounding for the downlink direction was performed by means of device measurements on **cell-specific reference signals** (**CRS**), which are transmitted over the whole transmission bandwidth in each subframe. From release 10 on, these reference signals were complemented by another type of reference signals called CSI-RS which, contrary to the first ones, are not expected to be continuously transmitted. The main reason for this upgrade lays in the necessity to support spatial multiplexing with more than four layers; anyway, as book [5] points out, CSI-RS introduction opened the way to further technological extensions, such has COrdinated Multi-Point operations and interference estimation.

5.1.1 CSI-RS structure

A CSI-RS may be associated to up to 32 different antenna ports, each port corresponding to a channel to be sounded [5]. To report the words used in the same book, an **antenna port** is such defined that the channel over which a symbol on the antenna port is conveyed can be inferred from the channel over which another symbol on the same antenna port is conveyed.

Reference [5] identifies two different classes of CSI-RS:

- a single-port CSI-RS occupies a single Resource Element (Resource Element (RE)) within a grid corresponding to one Resource Block (RB) in frequency domain and one slot in time domain; in principle, the RE can be selected anywhere in the resource block but, in practice, some restrictions exist to avoid collisions with other downlink physical channels and signals;
- a multi-port CSI-RS is defined as a group of orthogonally transmitted ports sharing a set of resource elements. The orthogonality property is achieved through a combination of multiplexing techniques:
  - Code Division Multiplexing (CDM): different per-antenna-port CSI-RS are transmitted on the same resource elements and they are separated by applying orthogonal modulation patterns;
  - Frequency Division Multiplexing (FDM): different per-antenna-port CSI-RS are transmitted on different subcarriers within the same OFDM symbols in the slot;
  - Time Division Multiplexing (TDM): different per-antenna-port CSI-RS are transmitted in different OFDM symbols within a slot.

5.1.2 CSI-RS mapping

The resource element pattern for an X-port CSI-RS spans N OFDM symbols in the same slot and it can consist of one ore more CSI-RS REs. A **resource element pattern** is defined within a single Physical Resource Block (PRB) as Y adjacent resource elements in the frequency domain and Z adjacent resource elements in the time domain. Some examples are shown in `Figure 5.1`.

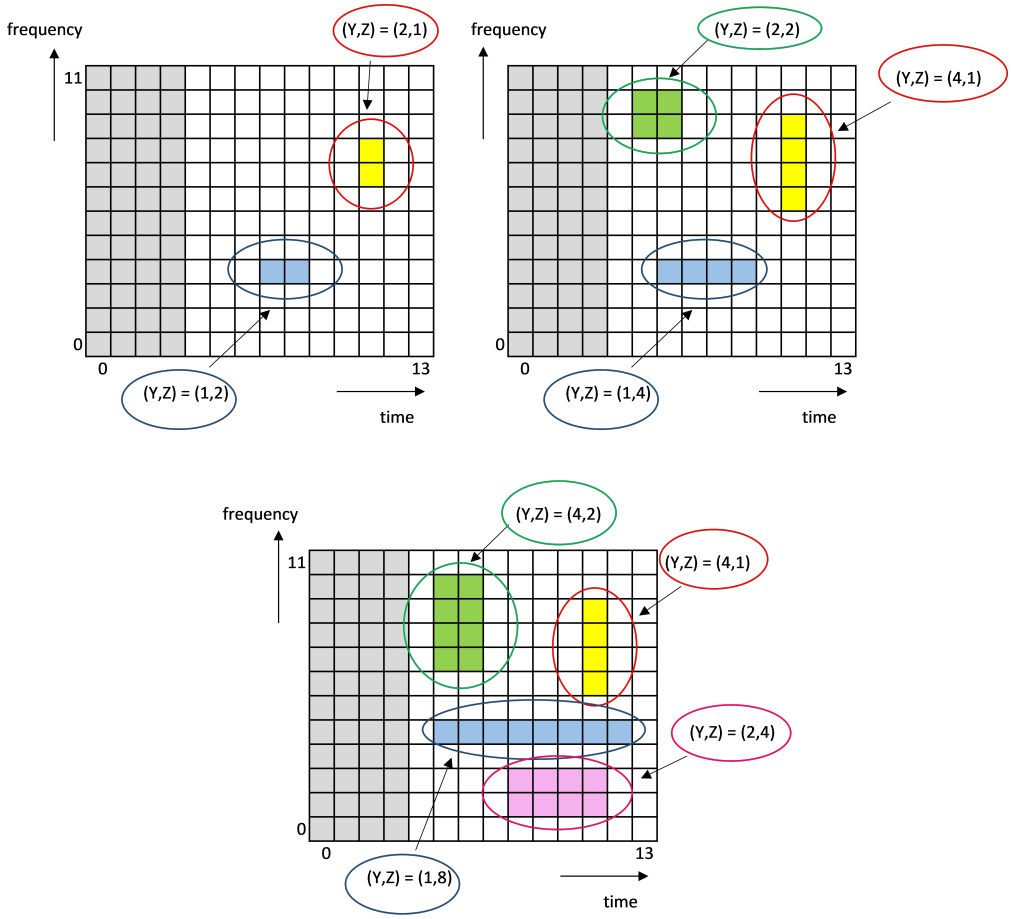


Figure 5.1: Examples of CSI-RS RE patterns

Reference [5] illustrates different CDM structures for multiplexing per-antenna-port CSI-RS which can be adopted:

- 2×CDM, which means CDM over two adjacent subcarriers, allowing for code-domain sharing between two per-antenna-port CSI-RS;
- 4×CDM, which means CDM over two adjacent subcarriers and two adjacent

OFDM symbols, allowing for code-domain sharing between up to four per-antenna-port CSI-RS;

- 8×CDM, which means CDM over two adjacent subcarriers and four adjacent OFDM symbols, allowing for code-domain sharing between up to eight per-antenna-port CSI-RS.

Row	Ports X	Density ρ	*cdm-Type*	$(\bar{k}, \bar{l})$	CDM group index j	k'	l'
1	1	3	No CDM	(k_0,l_0), (k_0+4,l_0), (k_0+8,l_0)	0,0,0	0	0
2	1	1, 0.5	No CDM	(k_0,l_0),	0	0	0
3	2	1, 0.5	FD-CDM2	(k_0,l_0),	0	0, 1	0
4	4	1	FD-CDM2	$(k_0,l_0),(k_0+2,l_0)$	0,1	0, 1	0
5	4	1	FD-CDM2	$(k_0,l_0),(k_0,l_0+1)$	0,1	0, 1	0
6	8	1	FD-CDM2	(k_0,l_0), (k_1,l_0), (k_2,l_0), (k_3,l_0)	0,1,2,3	0, 1	0
7	8	1	FD-CDM2	(k_0,l_0), $(k_1,l_0),(k_0,l_0+1)$, (k_1,l_0+1)	0,1,2,3	0, 1	0
8	8	1	CDM4 (FD2,TD2)	(k_0,l_0), (k_1,l_0)	0,1	0, 1	0, 1
9	12	1	FD-CDM2	(k_0,l_0), (k_1,l_0), (k_2,l_0), $(k_3,l_0),(k_4,l_0)$, (k_5,l_0)	0,1,2,3,4,5	0, 1	0
10	12	1	CDM4 (FD2,TD2)	(k_0,l_0), (k_1,l_0), (k_2,l_0)	0,1,2	0, 1	0, 1
11	16	1, 0.5	FD-CDM2	(k_0,l_0), (k_1,l_0), (k_2,l_0), $(k_3,l_0),(k_0,l_0+1)$, (k_1,l_0+1), (k_2,l_0+1), (k_3,l_0+1)	0,1,2,3, 4,5,6,7	0, 1	0
12	16	1, 0.5	CDM4 (FD2,TD2)	(k_0,l_0), (k_1,l_0), (k_2,l_0), (k_3,l_0)	0,1,2,3	0, 1	0, 1
13	24	1, 0.5	FD-CDM2	(k_0,l_0), (k_1,l_0), (k_2,l_0), (k_0,l_0+1), (k_1,l_0+1), $(k_2,l_0+1),(k_0,l_1)$, (k_1,l_1), (k_2,l_1), (k_0,l_1+1), (k_1,l_1+1), (k_2,l_1+1)	0,1,2,3,4,5, 6,7,8,9,10,11	0, 1	0
14	24	1, 0.5	CDM4 (FD2,TD2)	(k_0,l_0), (k_1,l_0), (k_2,l_0), (k_0,l_1), (k_1,l_1), (k_2,l_1)	0,1,2,3,4,5	0, 1	0, 1
15	24	1, 0.5	CDM8 (FD2,TD4)	(k_0,l_0), (k_1,l_0), (k_2,l_0)	0,1,2	0, 1	0, 1, 2, 3
16	32	1, 0.5	FD-CDM2	(k_0,l_0), (k_1,l_0), (k_2,l_0), $(k_3,l_0),(k_0,l_0+1)$, (k_1,l_0+1), (k_2,l_0+1), (k_3,l_0+1), (k_0,l_1), (k_1,l_1), (k_2,l_1), (k_3,l_1), (k_0,l_1+1), (k_1,l_1+1), (k_2,l_1+1), (k_3,l_1+1)	0,1,2,3, 4,5,6,7, 8,9,10,11, 12,13,14,15	0, 1	0
17	32	1, 0.5	CDM4 (FD2,TD2)	(k_0,l_0), (k_1,l_0), (k_2,l_0), (k_3,l_0), (k_0,l_1), (k_1,l_1), (k_2,l_1), (k_3,l_1)	0,1,2,3,4,5,6,7	0, 1	0, 1
18	32	1, 0.5	CDM8 (FD2,TD4)	(k_0,l_0), (k_1,l_0), (k_2,l_0), (k_3,l_0)	0,1,2,3	0,1	0,1, 2, 3

Figure 5.2: Table of CSI-RS location within a slot
(taken from [3])

These CDM alternatives, implemented together with FDM and/or TDM, give rise to different configurations of multi-port CSI-RS structures where, in general, an N-port CSI-RS occupies a total of $N \geq 1$ resource elements within a RB/slot [5]. The table in `Figure 5.2` reports the CSI-RS RE patterns for CSI acquisition that are presented in [3]: ρ is the density value measured in [RE/RB/port], k_0 and l_0 represent the time-frequency position of the pattern components while k' and l' are the indexes of the resource elements within a CSI-RS component. As it is possible to observe from this table, in the case of CSI-RS associated to more than two antenna ports, there are multiple CSI-RS configurations based on different combinations of CDM, TDM and FDM.

5.1.3 Frequency and time domain properties of CSI-RS

A CSI-RS is associated to a certain downlink **Bandwidth Part (BP)**: a bandwidth part is characterized by a numerology and by a contiguous set of physical resource

blocks (PRBs) on a given carrier [5]; these RBs are selected from a contiguous subset of the common resource blocks in that numerology. Each CSI-RS is confined within the correspondent bandwidth part and it is expected to adopt the numerology characterizing that specific bandwidth part. The CSI-RS can be configured in two different ways:

- covering the full bandwidth of the bandwidth part;
- covering just a fraction of the bandwidth part; in this case, the CSI-RS configuration has to provide the portion of bandwidth used and the starting position in the frequency domain.

Another flexible parameter to be configured is the CSI-RS **density** (ρ):

- density $\rho = 1$ implies that the CSI-RS transmission occurs in every resource block;
- density $\rho = \frac{1}{2}$ means that the CSI-RS transmission takes place every second resource block: in this case an additional information about the set of resource blocks used for CSI-RS transmission has to be provided;
- density $\rho = 3$ is a possible configuration for a single-port CSI-RS, which spans three subcarriers within each resource block.

Regarding the CSI-RS structure in the time domain, three different transmission modes are allowed:

- **periodic** transmission: CSI-RS are transmitted once every N slots;
- **semi-persistent** transmission: the device configuration determines a period and an offset, while the activation and the de-activation are controlled by means of MAC Control Element (MAC CE);
- **aperiodic** transmission: in this case, no periodicity exists and the transmission instant is signaled via Downlink Control Information (DCI).

5.1.4 CSI-RS mapping to physical antennas

As the authors of [5] explain, a multi-port CSI-RS corresponds to a set of antenna ports and the CSI-RS can be used for sounding the related channels. However, a CSI-RS port is often not directly mapped to a physical antenna: the CSI-RS may be subjected to many different types of transformation or spatial filtering, implying that the channel being sounded is not necessarily the actual physical radio channel. Anyway, book [5] clarifies that, when a device performs channel sounding based on the CSI-RS, both the

spatial filter and the physical antennas are completely transparent: what the device will see are just the N virtual channels corresponding to the N CSI-RS ports.

Basically there exist two types of CSI-RS to **Transceiver Unit (TXRU)** mapping:

- TXRU specific mapping: each CSI-RS is mapped to one TXRU, which in turn feeds several physical antenna elements;
- beamformed CSI-RS: each CSI-RS port is digitally virtualized to multiple TXRUs by means of a vector of beamforming weights; each TXRU in turn feeds several antenna elements.

Spatial Filtering

Although we have said that the **spatial filter** is totally transparent to the device, it remains an important feature, strongly connected to the concept of **antenna port**: as the authors of reference [5] make notice, the device can assume that two transmitted signals have experienced the same radio channel if and only if they are transmitted from the same antenna port. In essence, this can lead to the assumption that two signals are transmitted from the same antenna port if they are mapped to the same set of physical antennas by means of the same spatial filtering operation. Book [5] highlights the case of a downlink multi-antenna transmission: after performing CSI-RS-based channel measurements, a device may report a recommended **precoding matrix** to the network. The network in turn may decide to use the recommended precoding matrix to map the transmitted data streams, called transmission **layers**, to the antenna ports. However, whatever the matrix $\boldsymbol{W}$ chosen by the network, the device will assume that:

- the output of the precoding process will be mapped to the antenna ports of the CSI-RS on which the corresponding device measurements were taken;
- the precoded signal will be mapped to the physical antennas by means of the same spatial filter as the one applied to the CSI-RS.

5.2 Downlink feedback reporting

In 3GPP specifications [4], **Channel Quality Indicator (CQI)**, **Rank Indicator (RI)**, **Precoding Matrix Indicator (PMI)**, **CSI-RS Resource Indicator (CRI)**, Synchronization Signal/PBCH Block Resource Indicator (SSBRI) and Layer Indicator (LI) are jointly referred to as **Channel State Information** (**CSI**). The configuration of all these parameters for a UE is laid down by upper layers through $N \geq 1$ *CSI-ReportingConfig* Reporting Settings and $M \geq 1$ *CSI-ResourceConfig* Resource Settings.

5.2.1 CSI reporting types

The time domain behavior of the CSI-RS resources within a CSI Resource Setting is determined by a higher layer parameter and it can be classified as **aperiodic**, **periodic**, or **semi-persistent**:

- periodic reporting is performed with a certain periodicity, always on the Physical Uplink Control CHannel (PUCCH); in this case the resource configuration also includes information about the Physical Uplink Control CHannel (PUCCH) employed for periodic reporting;
- in semi-persistent reporting, periodic reporting instances are configured in the same way as for periodic reporting; however, the real reporting procedure can be enabled or disabled by means of Medium Access Control (MAC) signaling and the resource allocated for reporting can be a periodic PUCCH resource or a semi-persistently allocated Physical Uplink Shared CHannel (PUSCH);
- aperiodic reporting is triggered by means of Downlink Control Information (DCI) within a CSI-request field inside the uplink scheduling grant; the DCI field may consist of up to 6 bits and each specific bit combination corresponds to a particular aperiodic report.

5.3 CSI measurements and reporting for downlink multi-antenna precoding

In book [5], the authors define **layer-mapping** as the step to distribute the modulation symbols across the different transmission layers. Similarly to LTE, the n-th symbol is mapped to the n-th layer. One coded transport block can be mapped on up to four layers so, when downlink transmissions support from five to eight layers, a second transport block is mapped to the layers from five to eight [5].
The authors point out that multi-layer transmission is only allowed in downlink: since in uplink a DFT-precoding is performed, only a single transmission layer is supported, due to prohibitive complexity requirements at the receiver.

The purpose of **multi-antenna precoding** is to map the different transmission layers to a set of antenna ports using a **precoding** or **beamforming matrix** (**W**) [5]. Any downlink multi-antenna precoding operation is transparent to the device, since the Demodulation Reference Signals (DM-RS) used for coherent channel estimation at the user side are multiplied by the same precoding matrix applied at the transmitter. This transparency implies that the network can in principle apply any beamforming, without any impact on device procedures.

As it is remarked in [5], the 3GPP specification of downlink multi-antenna precoding is mainly related to the measurements and reporting performed by the device to support the selection by the network of a precoder for downlink PDSCH transmission. These measurements and reporting steps are within the scope of the more general CSI reporting framework, as clarified in the previous section. For this purpose, **precoder codebooks** have been defined in the standard.

5.3.1 CSI-based precoder codebook

As already mentioned, the Precoding Matrix Indicator reported by the UE is an indication on what the user considers a suitable precoding matrix to be applied on downlink transmissions. In particular, the PMI is nothing more than a set of indexes or a single index pointing to a specific entry in the precoder codebook, which contains a list of all the possible precoder matrices $\boldsymbol{W}$ that the device can select and report to the network. There exists at least one codebook for each permitted combination of N_T (number of antenna ports) and N_L (number of layers), associated with the configured CSI-RS.
An important consideration must be made: precoder codebooks play a role only in the context of PMI reporting and they do not impose any restriction in the choice of the precoding matrix to be applied by the network [5]. In some cases it is convenient for the network to select the beamforming matrix suggested by the device but, when for example the network is in possession of additional information, the precoding computation may lead to a different choice.

As book [5] reports, typical use case of multi-antenna precoding is **Multi-User MIMO** (MU-MIMO): in this case the main purposes are to direct the energy towards the device and simultaneously limiting the interference towards the other users scheduled on the same time-frequency resource. In this kind of scenario, it is clear that the network needs a more detailed information about the channel experienced by different UEs, so that it can take into account the PMI reported by all simultaneously scheduled devices when selecting the precoding matrix [5].

NR standard defines two types of CSI reporting, each characterized by a different size and structure of the codebook:

- **Type I CSI codebooks**: they are designed for Single User-MIMO (SU-MIMO) scenarios, where a single user is scheduled within a certain time-frequency resource, transmitting on a potentially large number of layers in parallel;
- **Type II CSI codebooks**: they are designed for MU-MIMO scenarios, where many users are scheduled within the same time-frequency resource, each transmitting on a limited number of layers in parallel (one or two).

5.3.2 Type I CSI codebooks

As it is explained in [5], the main aim of these codebooks is to focus the energy towards the target device; the interference caused by the adoption of a potential high number of layers in parallel is not managed since it is assumed to be handled at the receiver side through multi-antenna processing. Type I codebooks are further divided into two sub-types, corresponding to different antenna configurations on the network side:

- **single panel antennas CSI**;
- **multi-panel antennas CSI**.

Type I - Single panel codebook

The expression *single panel* refers to those antenna configurations in which the $N_1 \times N_2$ cross-polarized antenna elements are located on a single panel. In this scenarios, the precoding matrix $\boldsymbol{W}$ is generated as the product of matrices $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$:

$$\boldsymbol{W} = \boldsymbol{W_1} \cdot \boldsymbol{W_2} \tag{5.1}$$

where:

$$\boldsymbol{W_1} = \begin{bmatrix} \boldsymbol{B} & 0 \\ 0 & \boldsymbol{B} \end{bmatrix} \tag{5.2}$$

and

$$\boldsymbol{B} = \begin{bmatrix} b_0 & b_1 & ... & b_{L-1} \end{bmatrix} \tag{5.3}$$

Each vector $\boldsymbol{B}$ contains L neighbor beams, where each beam is used for both the polarizations. Thus, the selection of a matrix $\boldsymbol{W_1}$ coincides with the selection of a specific beam direction from all the possible beam directions [5].

In order to deepen the generation process of the matrix $\boldsymbol{W_1}$, it is necessary to introduce the concept of **DFT oversampling**: as a result of the oversampling, `Figure 5.3` shows intermediate beams added to the four original orthogonal beams defined by the matrix $\boldsymbol{W_1}$. The oversampling operation affects both vertical and horizontal direction, in a way that is determined by two oversampling factors, one for the horizontal direction (O_1) and the other for the vertical one (O_2). In this specific example, both $O_1 = 4$ and $O_2 = 4$, implying $N_1 \cdot O_1$ beams in the horizontal direction and $N_2 \cdot O_2$ beams in the vertical direction, where N_1 and N_2 correspond to the number of antenna elements for each dimension of the panel. `Figure 5.4` reports all the configurations of (N_1, N_2) and (O_1, O_2) supported in Release 15.

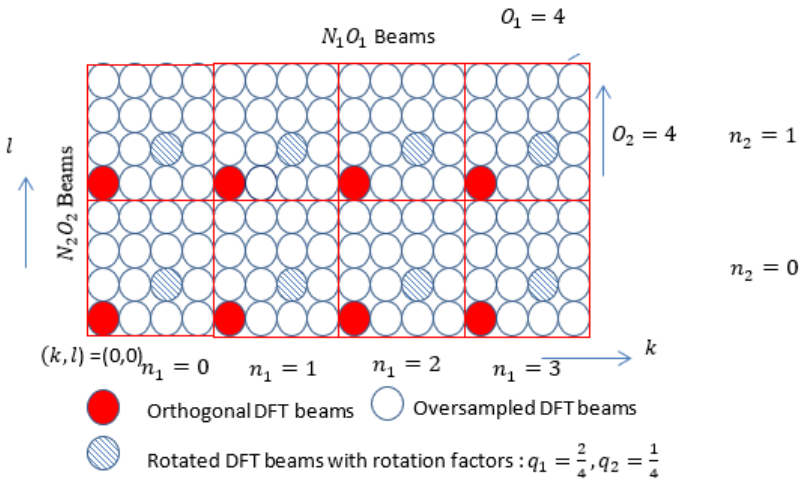


Figure 5.3: Example of oversampled beams
(taken from [W2])

Number of CSI-RS antenna ports, $P_{\text{CSI-RS}}$	(N_1, N_2)	(O_1, O_2)
4	(2,1)	(4,1)
8	(2,2)	(4,4)
	(4,1)	(4,1)
12	(3,2)	(4,4)
	(6,1)	(4,1)
16	(4,2)	(4,4)
	(8,1)	(4,1)
24	(4,3)	(4,4)
	(6,2)	(4,4)
	(12,1)	(4,1)
32	(4,4)	(4,4)
	(8,2)	(4,4)
	(16,1)	(4,1)

Figure 5.4: Supported configurations of (N_1, N_2) and (O_1, O_2)
(taken from [4])

Figure 5.5 illustrates the construction process of $\boldsymbol{W_1}$, leaving room for some observations:

- the matrix $\boldsymbol{B}$, defined in (5.3), can be seen as the kronecker product between $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$;
- $\boldsymbol{X_1}$ is a matrix of N_1 columns, each containing the horizontal beamforming weights for the antenna elements on the correspondent row;
- $\boldsymbol{X_2}$ is a matrix of N_2 columns, each containing the vertical beamforming weights for the antenna elements on the correspondent column;
- each column of $\boldsymbol{W_1}$ represents the weight vector for each specific beam formed by the antenna array.

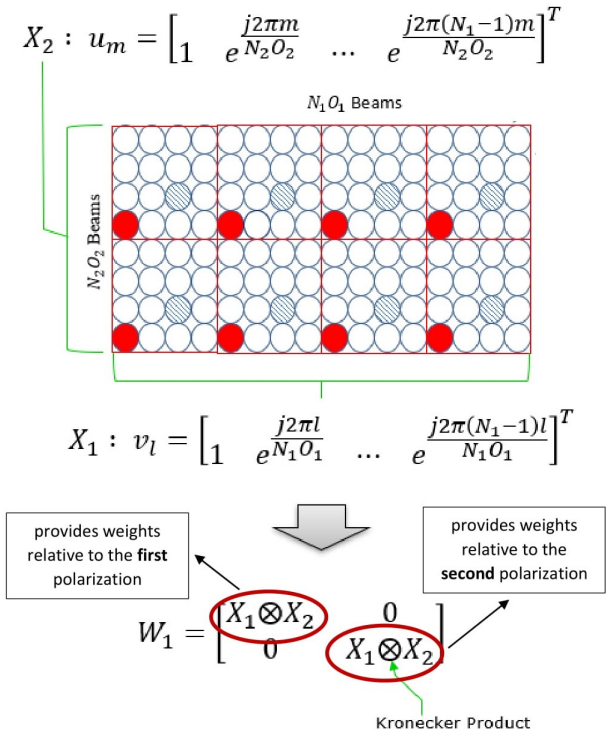


Figure 5.5: Generation process of W_1 matrix (taken from [W3])

In the case of transmissions with rank 1 or 2, the matrix W_1 can define a single beam or a group of four neighbor beams: if four beams are identified by the matrix W_1, with the matrix W_2 a further selection among the four beams (corresponding to W_1 columns) is performed, together with the identification of the QPSK co-phasing of the two polarizations; if instead a single beam is defined by the matrix W_1, the matrix W_2 simply provides co-phasing between the two polarization [5]. The table in `Figure 5.6` shows the codebooks for 1-layer and 2-layer CSI reporting using two antenna ports. In the case of transmissions with rank $R > 2$, the matrix W_1 collects $N = \frac{R}{2}$ orthogonal beams, while the matrix W_2 only provides co-phasing between the two polarizations.

According to the standard, up to eight layers can be transmitted towards the same device, exploiting the four beams selectable by the matrix W_1 and the two polarizations. To summarize:

- matrix W_1 gathers long-term frequency-independent channel characteristics and thus is reported on wideband basis;
- matrix W_2 instead captures short-term frequency-dependent channel characteristics and thus it is specific for each subband.

Codebook index	Number of layers υ	
	1	2
0	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{2}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
1	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$	$\frac{1}{2}\begin{bmatrix}1 & 1\\j & -j\end{bmatrix}$
2	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	-
3	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-j\end{bmatrix}$	-

Figure 5.6: Codebooks for 1-layer and 2-layer CSI reporting
(taken from [4])

Type I - Multi-panel codebook

These kind of codebooks are designed for scenarios where multiple antenna panels are jointly used at the network side. In particular, the configuration includes N_g 2D-panels ($N_g \in \{2,4\}$), each with $N_1 \times N_2$ cross-polarized antenna elements.
As explained in book [5], the basic principle of Type I multi-panel CSI is the same as that of single-panel, with some notable differences:

- the matrix $\boldsymbol{W_1}$ defines one beam per polarization and panel;
- the matrix $\boldsymbol{W_2}$ provides per-subband co-phasing between polarizations and panels;
- the maximum number of layers supported for spatial multiplexing goes down to four.

Number of CSI-RS antenna ports, $P_{\text{CSI-RS}}$	(N_g, N_1, N_2)	(O_1, O_2)
8	(2,2,1)	(4,1)
16	(2,4,1)	(4,1)
	(4,2,1)	(4,1)
	(2,2,2)	(4,4)
32	(2,8,1)	(4,1)
	(4,4,1)	(4,1)
	(2,4,2)	(4,4)
	(4,2,2)	(4,4)

Figure 5.7: Supported multi-panel configurations of (N_g, N_1, N_2) and (O_1, O_2)
(taken from [4])

The table in Figure 5.7 shows the configurations of 1D/2D antenna port layout (N_g, N_1, N_2) and oversampling factors (O_1, O_2) supported.

5.3.3 Type II CSI codebooks

Similarly to Type I CSI, Type II CSI is based on wideband selection and reporting of beams, but there exists some significant differences, highlighted in [5]:

- while Type I CSI selects and report one single beam, Type II CSI can select and report up to four orthogonal beams with their corresponding amplitude and phase values;
- the reported precoding matrix from Type II CSI has non-constant modulus, which means that different elements in the matrix $\boldsymbol{W}$ can present different amplitudes;
- since the typical usage scenarios are MU-MIMO, in which the network identifies a group of transmissions to schedule on the same time-frequency resources and the corresponding precoding matrices, the number of layers per device is limited to two.

To conclude, it is clear that this second type of reporting provides much more detailed information about the channel, with significantly higher spatial granularity [5].

5.4 Deep Learning-based CSI feedback for Massive MIMO scenarios

When working in FDD mode, the UE estimates the downlink Channel State Information and reports it to the base station through feedback links: this information is exploited at the network side to obtain gains for Massive MIMO transmissions. As remarked in paper [23], when the number of antennas in transmission and reception considerably increases, the feedback overhead may become excessive. To solve this issue, the authors propose a Deep Learning-based CSI sensing and recovery mechanism, named CsiNet. This is a very innovative proposal since other relevant works treated DL-based approaches for CSI encoding, but none of them considered the recovery stage.

5.4.1 CsiNet model

The CSI Network (CsiNet) model developed by the authors of paper [23] consists of:

- an encoder, that learns a mapping from the channel matrices to the compressed codewords;
- a decoder, that learns the inverse mapping, from the compressed codewords to the original CSI.

Summarizing, the UE applies the encoder function to transform the channel matrices into codewords, while at the BS the decoder is used to recover the original CSI information.

System model

In paper [23], a MIMO system with $N_t >> 1$ transmit antennas at the BS and a single receiver antenna at the UE is considered. The received signal on the n-th subcarrier is expressed as

$$y_n = \hat{\boldsymbol{h}}_n^H \boldsymbol{v}_n x_n + z_n \tag{5.4}$$

where

- $\hat{\boldsymbol{h}}_n$ is the complex channel vector of size $N_t \times 1$;
- $\boldsymbol{v_n}$ is the complex precoding vector of size $N_t \times 1$;
- x_n is the complex data symbol received on the n-th subcarrier;
- z_n denotes the additive noise.

The precoding vector $\boldsymbol{v} = \{\boldsymbol{v}_n : n = 1, ..., N_c\}$ is selected by the BS once it receives the CSI feedback information $\tilde{\boldsymbol{H}} = [\tilde{\boldsymbol{h}}_1, ..., \tilde{\boldsymbol{h}}_{\tilde{N}_c}]^H \in \mathbb{C}^{\tilde{N}_c \times N_t}$. Thus, the total number of feedback values amounts to $\tilde{N}_c N_t$, which in some cases could be an excessive quantity.

DFT-based preprocessing of CSI information

For their experiments, the authors of paper [23] assume that perfect channel matrices are available through pilot-based estimation techniques, completely neglecting the downlink channel estimation stage and focusing on the feedback procedure. Their approach includes a 2D-DFT operation to sparsify the matrix $\tilde{\boldsymbol{H}}$ in the angular-delay domain as follows:

$$\boldsymbol{H} = \boldsymbol{F_d} \tilde{\boldsymbol{H}} \boldsymbol{F_a^H} \tag{5.5}$$

where $\boldsymbol{F_d}$ and $\boldsymbol{F_a}$ are $\tilde{N}_c \times \tilde{N}_c$ and $N_t \times N_t$ DFT matrices respectively. The reason behind this processing is that only a small portion of the matrix $\boldsymbol{H}$ contains large values, while a consistent portion of it consists of elements close to zero: thinking to the fact that the

interarrival time between multipaths lies within a limited interval, it is clear that only the first $N_c < \tilde{N}_c$ rows of $\boldsymbol{H}$ are relevant and the remaining ones can be removed. So, after performing a truncation operation, $\boldsymbol{H}$ becomes a matrix of size $N_c \times N_t$, implying a reduction of the feedback size. The truncated matrix is first scaled to range [0,1] and then it is encoded into an M-dimensional codeword, with $M < N_c N_t$, providing a compression ratio equal to $\gamma = \frac{M}{N_c N_t}$. In a specular way, at the output of the decoder implemented at the BS an inverse DFT operation has to be performed.

CsiNet encoder

The CsiNet encoder is essentially a deep CNN, whose input consists of the real and imaginary parts of the truncated matrix $\boldsymbol{H}$. The first layer is a convolutional layer with filter size equal to $3 \times 3 \times 2$, that generates two feature maps. Then, a reshaping operation is performed to vectorize the two feature maps. Finally, a fully connected layer generates a vector (codeword) $\boldsymbol{s}$ of size M.

CsiNet decoder

Also the CsiNet decoder is a CNN, whose input consists of the codeword $\boldsymbol{s}$ generated by the encoder and sent to the BS. The first layer is a fully connected layer that outputs two $N_c \times N_t$ matrices, representing an initial estimation of the real and imaginary parts of the matrix $\boldsymbol{H}$. Then, the initial estimates are fed into some *RefineNet* units, whose purpose is to progressively refine the reconstruction. Each RefineNet unit is composed by an input layer and three convolutional layers using a kernel of size 3×3: the second layer generates 8 feature maps, the third 16 while the fourth outputs the final reconstruction of $\boldsymbol{H}$. The feature maps produced by the three layers of a RefineNet unit are forced by a zero-padding technique to have the same size as the input channel matrix ($N_c \times N_t$). Moreover, each layer is followed by a ReLU and a Batch Normalization layer. Experiments lead the authors to conclude that two RefineNet units are sufficient to obtain good performance; adding further RefineNet units would not significantly impact the reconstruction quality, at the cost of a larger complexity. After the RefineNet units, a final convolutional layer is placed and its output is then fed into a softmax layer that scales the values to the [0,1] range.

Training process

In paper [23], the set of trainable parameters is denoted as $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_{en}, \boldsymbol{\Theta}_{de}\}$, while the input and the output of the CsiNet model for the i-th patch are denoted as $\boldsymbol{H}_i$ and $\hat{\boldsymbol{H}} = f(\boldsymbol{H}_i; \boldsymbol{\Theta}) = f_{de}(f_{en}(\boldsymbol{H}_i; \boldsymbol{\Theta}_{en}); \boldsymbol{\Theta}_{de})$ respectively.

The loss function is defined as

$$L(\boldsymbol{\Theta}) = \frac{1}{T}\sum_{i=1}^{T} ||f(\boldsymbol{s}_i;\boldsymbol{\Theta}) - \boldsymbol{H}_i||_2^2 \tag{5.6}$$

where T denotes the number of training samples.

Simulation results

To test the model performance, the authors of paper [23] generate training and testing datasets from the COST 2100 channel model for the indoor picocellular scenario at 5.3 GHz and for the outdoor rural scenario at 300 MHz.

Table 5.1: CsiNet hyperparameters setting

Hyperparameter	Configured value
learning rate	0.001
batch size	200
iterations	1000
training samples	100000
validation samples	30000
testing samples	20000

The number of antennas at the BS is set to $N_t = 32$, while the number of subcarriers used is set to $\tilde{N}_c = 1024$; these 1024 subcarriers are reduced to 32 once the angular-domain transformation has been performed, so that $\boldsymbol{H}$ results to be a 32×32 matrix. As regards CsiNet hyperparameters configuration, the selected values are reported in `Table 5.1`.

The authors dedicates a section of the paper to the comparison of CsiNet performance with three traditional CS-based methods, namely LASSO l_1 solver, TVAL3 and BM3D_AMP, that are not deepened in this section. Performances are evaluated in terms of:

- NMSE, computed as

$$NMSE = E\{\frac{||\boldsymbol{H} - \hat{\boldsymbol{H}}||_2^2}{||\boldsymbol{H}||_2^2}\} \tag{5.7}$$

  that measures the difference between the recovered channel matrix $\hat{\boldsymbol{H}}$ and the original channel matrix $\boldsymbol{H}$;

- cosine similarity, computed as

$$\rho = E\{\frac{1}{\tilde{N}_c}\sum_{n=1}^{\tilde{N}_c}\frac{|\hat{\tilde{\boldsymbol{h}}}_n^H \tilde{\boldsymbol{h}}_n|}{||\hat{\tilde{\boldsymbol{h}}}_n||_2||\tilde{\boldsymbol{h}}_n||_2}\} \tag{5.8}$$

  that measures the quality of the beamforming vector defined as $v_n = \hat{\tilde{\boldsymbol{h}}}_n/||\hat{\tilde{\boldsymbol{h}}}_n||_2$ at the user side.

Simulation results show that, compared to the other CS-based algorithm, the CsiNet solution provides significant gain and outperforms them at all compression ratios. Moreover, CsiNet recovery process can work faster with respect to the CS-based approaches, since it consists of simple matrix-vector multiplication steps.

Finally, the authors conclude with two important observations:

- experiments show that CsiNet can achieve good performance even if the DFT-matrix F_a, which transforms the matrix $\tilde{\boldsymbol{H}}$ from the spatial to the angular domain, is not applied; this means that the DL-based model can learn proper basis without preprocessing the channel matrix, making this solution very flexible;

- from the results obtained it is clear that, increasing the number of antenna at the BS N_t, the reconstruction performances of all the algorithms improve: according to paper [23], this happens since an increase of N_t implies an increase of the spatial resolution and so the matrix $\boldsymbol{H}$ becomes sparser.

To conclude, the work presented in paper [23] opens the way to further improved DL-based mechanisms for CSI sensing and recovery: in fact, in the light of the promising results obtained, the authors themselves encourage future research in this direction. Given that, the following section investigates the application of CsiNet solution to New Radio systems: besides adapting the model to the new communication environment, we experiment some variations of the original algorithm proposed in the paper, with the purpose of enhancing its performance and its applicability in more practical contexts.

5.5 A CsiNet model for NewRadio

In this section, we introduce a DL-based solution for CSI information reduction and recovery in a 5G context. In particular, we implement a deep CNN, named **NR-CsiNet**, which substantially inherits the same structure of the model presented in `Section 5.4`, but with some variations and additions that aim at generalizing the approach to a wider and more practical range of use cases:

- first of all, we try to make the model applicable also to multi-receiver antenna scenarios, overcoming the restriction imposed by paper [23], in which the study is limited to single receiver antenna communications. Substantially, instead of considering only 2D-CSI images, our model has to deal with a third dimension, representing the different receiver antennas. To manage this additional information we propose two alternative approaches, which will be described in the following subsections;

- another important novelty introduced by this thesis project is the implementation of a Variational AutoEncoder (VAE), which is presented in one of the following subsections. As it is confirmed in reference [6], VAEs have recently emerged as one of the most popular approaches to unsupervised learning of complex distributions, since they have demonstrated a better ability to compress complicated datasets into simpler mainfolds;

- finally, whereas downlink channel estimation is beyond the scope of reference [23], we consider also this challenging topic. In fact, the authors of the paper focus uniquely on the feedback procedure, assuming to acquire perfect CSI information, while we also deal with the CSI estimation. This means that, instead of feeding our NR-CsiNet with perfect CSI images, we give in input noisy images estimated through the CSI-RS, introduced in Section 5.3.

Also in this case, we used the Tensorflow's Keras API to build and train our autoencoder. Inside the link level simulator that we have already presented in Subsection 4.4, we substituted the traditional feedback reporting block with our deep learning-based alternative: this new block imports the pre-trained Keras encoder and uses it to elaborate an encoded version of the CSI matrix, previously estimated from the CSI-RS. Then, at the transmitter side, we inserted a specular decoding block which imports the pre-trained decoder model and uses the reconstructed CSI matrix at its output to determine the optimal beamforming vector.

5.5.1 NR simulator parameters

As for NR-ChannelNet, also for NR-CsiNet experiments we use the NR link simulator presented in Section 4.4 to collect training data and to evaluate the performance of the model. Table 5.2 reports the configuration of the simulator parameters that are of particular interest in our experiments.

As regards the transmitter and receiver antenna systems, we consider a single configuration, showed in Table 5.3.

Table 5.2: Simulation parameters for NR-CsiNet training

Simulation parameters	
Transmission direction	Downlink
System bandwidth	5 MHz
Subcarrier Spacing	15 kHz
TTI duration	1 ms
Carrier frequency	3.64 GHz
iMCS	21
(Modulation, Coding rate)	(64-QAM, 0.694)
TBS	19968 bit
NFFT	512
NSUB	300
NPRB	25
CP Type	Normal
NOFDM	14
Number of transmission layers	1
Number of codewords	1
DM-RS Type	Type 1
CSI-RS period	4 TTIs
CSI-RS beamforming	OFF
Channel model	CDL-B
Delay Spread model	Nominal

Table 5.3: Antenna systems configurations considered for NR-CsiNet training

Antenna system configuration					
Configuration	N_{TX}	N_{RX}	N_1	N_2	N_{pol}
8×2	8	2	2	2	2

N_{TX} and N_{RX} stand for the number of transmitter and receiver antennas respectively, while N_1 and N_2 represent the number of antenna elements on the azimuth and elevation plane; N_{pol} instead indicates the number of polarizations for each antenna element.

5.5.2 NR-CsiNet for CSI feedback reporting

Our NR-CsiNet model basically implements an autoencoder, whose purpose is to learn a different representation of the input data and the corresponding inverse function to reconstruct the original data starting from the transformed ones. Distancing from the standard, which includes two different types of CSI feedback reporting (`Section 5.3`), our CSI autoencoder works as follow:

- at the UE side, the CSI-encoder transforms the downlink channel matrix into compressed vectors that are reported to the BS;
- at the network side, starting from the compressed feedback received, the CSI-decoder reconstructs the original downlink CSI feedback, so that the BS can exploit this information to select the proper beamforming vector.

To summarize, if we focus uniquely on the compression task, ignoring the downlink channel estimation step, what our model is expected to do is to take in input a clean image of the channel and learn efficient encoding and decoding algorithms to compress and decompress it with the minimum error.

Training dataset

The channel matrix that we consider for our NR-CsiNet, named $\boldsymbol{H_{CSI,id}}$, is different from the channel image considered for the NR-ChannelNet model ($\boldsymbol{HW_{id}}$):

- first of all, $\boldsymbol{H_{CSI,id}}$ is not a time-frequency grid, since it is relative to one single OFDM symbol and it is generated by the CDL channel model once every CSI-RS period;
- secondly, the matrix $\boldsymbol{H_{CSI,id}}$ does not represent the beamformed channel: setting the simulator parameter `CSI-RS beamforming` to `OFF` (`Table 5.2`), the CSI-RS symbols are not subject to any beamforming operation. For this reason, even when considering a single receiver antenna, the channel information about distinct transmitter antennas cannot be collapsed in one single vector.

While the authors of paper [23] limit their DL-based solution to single receiver antenna scenarios, our purpose is to make our model applicable also in MIMO scenarios. With this in mind, the downlink channel information relative to each distinct receiver antenna must be in some way organized within our CSI matrix $\boldsymbol{H_{CSI,id}}$. In particular the matrix $\boldsymbol{H_{CSI,id}}$ must have a dedicated dimension for each receiver antenna and turns out to be a $NSUB \times N_{TX} \times N_{RX}$ complex matrix, as summarized in `Figure 5.8`.

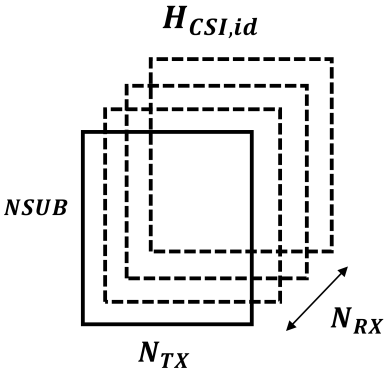


Figure 5.8: CSI matrix for CsiNet model input

To deal with the addition of a third dimension in the input CSI information, we decide to adopt the same two approaches followed for the NR-ChannelNet model and described in Subection 4.5.1:

1. single-channel approach: CSI images relative to distinct receiver antennas are fed to the networks on the same input channel;

2. multi-channel approach: each receiver antenna dimension is translated into a distinct input channel for the CsiNet model.

It is important to observe that, differently from the NR-ChannelNet model presented in Section 4.5, this Deep Learning approach does not require $\{input, output\}$ pairs: the desired output is indeed the input itself, since our NR-CsiNet autoencoder is expected to reconstruct the CSI original information as faithfully as possible. Considering this, the DL-based algorithm at the basis of our NR-CsiNet model can be classified as unsupervised.

Data preprocessing: an overview on compressive sensing

The NR-CsiNet model presented in this section requires a more sophisticated data preprocessing procedure than the one illustrated in Subsection 4.5.1. In this case interpolation is not necessary, since the complete and clean CSI image $\boldsymbol{H_{id,CSI}}$ is available. In Figure 5.9 it is possible to observe an example of perfect CSI matrix, with $NFFT = 1024$, $NSUB = 600$ and $N_{TX} = 8$.

Instead, to reduce feedback overhead, the authors of paper [23] propose a 2D-Discrete Fourier transform (DFT) transformation, with the purpose of sparsifying the information in the angular-delay domain. Since in our NR-CsiNet model, we consider the case of multiple receiver antennas, the 2D-DFT operation must be

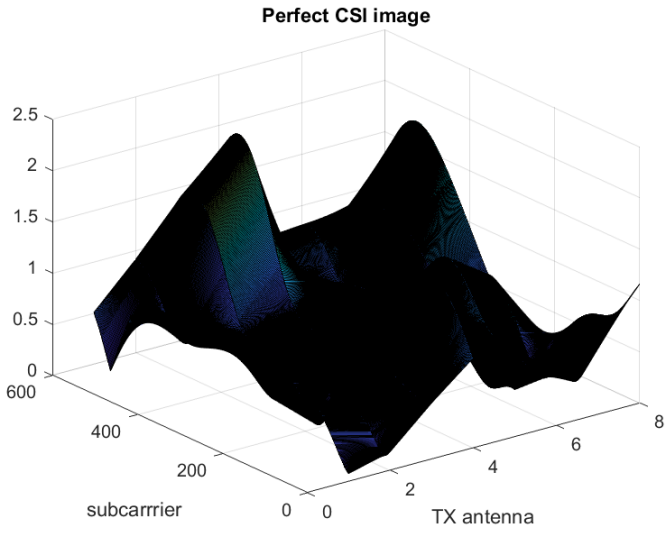


Figure 5.9: Absolute value of perfect CSI image $\boldsymbol{H_{id,CSI}}$

performed for each $NSUB \times N_{TX}$ matrix $\boldsymbol{H^i_{id,CSI}}$, where the apex $i \in [1, N_{RX}]$ represents the index of the receiver antenna that we are considering.

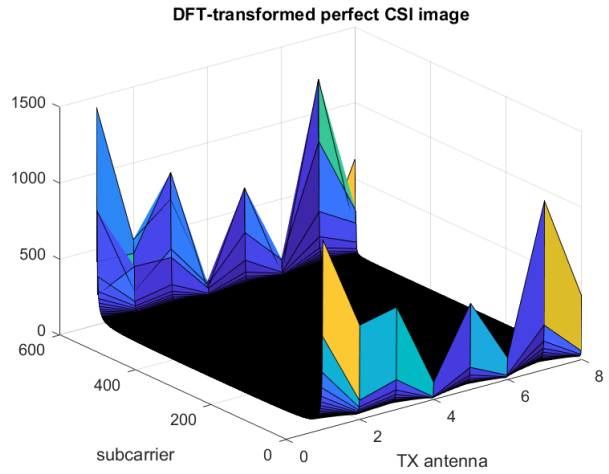


Figure 5.10: Absolute value of DFT-transformed $\boldsymbol{H^{DFT,i}_{id,CSI}}$ matrix

With this transformation, we get the sparsified matrix $\boldsymbol{H^{DFT,i}_{id,CSI}}$ (`Figure 5.10`), which can be expressed as follows:

$$\boldsymbol{H^{DFT,i}_{id,CSI}} = \boldsymbol{F_d} \boldsymbol{H^i_{id,CSI}} \boldsymbol{F_a} \tag{5.9}$$

where $\boldsymbol{F_d}$ is the DFT $NSUB \times NSUB$ matrix which performs the 1D-DFT transformation of all the rows, while $\boldsymbol{F_s}$ represents the DFT $N_{TX} \times N_{TX}$ matrix which performs the

1D-DFT transformation of all the columns.

To understand the utility of this approach, it is necessary to make a digression about compressive sensing (CS) techniques and the related concept of sparsity. As it is explained in paper [15], CS encoding is a technique capable of capturing the K non-zero components of a K-sparse $N \times 1$ signal vector, where $K << N$, by compressing the signal into $M << N$ measurements via random projections. A signal or image $\boldsymbol{x} \in \mathbb{R}^N$ is defined as K-sparse if only K coefficient $< \boldsymbol{x}, \psi_i >$ are non-zero, where $\{\psi_i \in \mathbb{R}^N\}_i = 1^n$ and $\boldsymbol{\Psi} = [\psi_1, ..., \psi_N] \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$. The set of ψ_i represents an orthogonal basis for $\mathbb{R}^N$. However, in many practical cases, the starting vector $\boldsymbol{x}$ does not exhibit any sparsity by itself and so a sparsifying preprocessing step has to be performed; in matrix notation, this becomes

$$\boldsymbol{s} = \boldsymbol{\Psi x} \tag{5.10}$$

where $\boldsymbol{s}$ denotes the sparse representation of $\boldsymbol{x}$, and $\boldsymbol{\Psi}$ is an $N \times N$ sparsifying-basis. The matrix $\boldsymbol{\Psi}$ must be selected so that, expressing the signal or the image in the form $\boldsymbol{x} = \sum_{i=1}^{N} \psi_i < \boldsymbol{x}, \psi_i >$, large coefficients are relatively few while small coefficients are a lot. A typical choice for Ψ is precisely the Discrete Fourier transform (DFT) matrix. At this point, in CS scenarios, the sparse signal representation $\boldsymbol{s}$ can be compressed into an $M \times 1$ vector denoted as $\boldsymbol{y}$, according to the following expression:

$$\boldsymbol{y} = \boldsymbol{\Phi s} \tag{5.11}$$

where $\boldsymbol{\Phi}$ is a $M \times N$ measurement matrix with independent and identically distributed random entries, which can be generated in accordance to a Gaussian or a Bernoulli distribution.

The DFT-based method proposed in paper [23] is not exactly a CS technique, because it simply sparsifies the matrices $\boldsymbol{H}^{i}_{\boldsymbol{id,CSI}}$ in the angular-delay domain, exploiting the correlation between transmitter antennas: in fact, as time delays between multipath arrivals range within a limited period, only a small number of rows in the matrix $\boldsymbol{H}^{\boldsymbol{DFT,i}}_{\boldsymbol{id,CSI}}$ ends up having non-zero values, as `Figure 5.10` shows. Looking at the picture, it is clear that a DFT-shift is necessary to concentrate the peaks in the same area of the graph: the result of this procedure is shown in `Figure 5.11`.

At this point, a simple truncation of the matrix, consisting in taking only the $NSUB/K_{DFT} < NSUB$ central rows, corresponding to the smaller delays values, and eliminating all the side points with almost zero values. The parameter denoted as K_{DFT} represents the compression factor with respect to the original number of subcarriers $NSUB$. `Figure 5.12` displays the result that we get by considering only the i-th receiver antenna, that is the matrix $\tilde{\boldsymbol{H}}^{\boldsymbol{DFT,i}}_{\boldsymbol{id,CSI}}$ of size $NSUB/K_{DFT} \times N_{TX}$. By means of an inverse

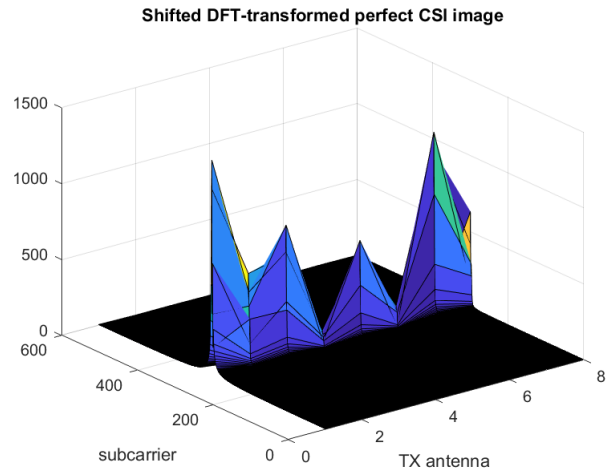


Figure 5.11: Absolute value of shifted $\boldsymbol{H}_{\boldsymbol{id,CSI}}^{\boldsymbol{DFT,i}}$ matrix

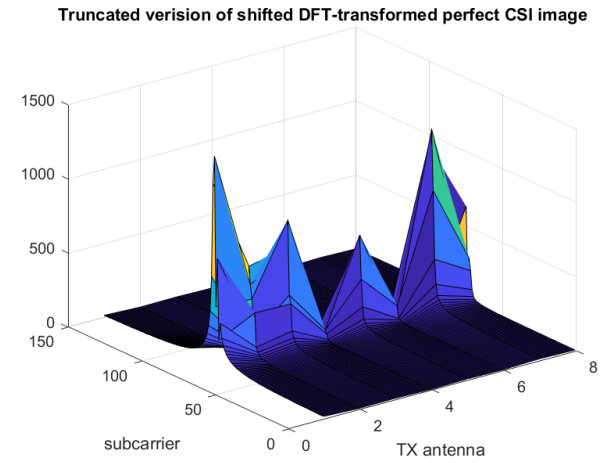


Figure 5.12: Absolute value of truncated $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}^{\boldsymbol{DFT,i}}$ matrix, with $K_{DFT} = 4$

DFT-shift and an inverse DFT operation (IDFT), each matrix $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}^{\boldsymbol{DFT,i}}$ is then brought back in the spatial-frequency domain and the resulting matrices are denoted as $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}^{\boldsymbol{IDFT,i}}$. The reason behind this step, that falls outside the preprocessing phase described in [23], lies in the fact that in the spatial-frequency domain the CSI images present a smoother profile, as it can be observed in `Figure 5.13`; images that present lines rather crisp and sharp, as the one showed in `Figure 5.12`, are generally more difficult to compress and reconstruct for a CNN.

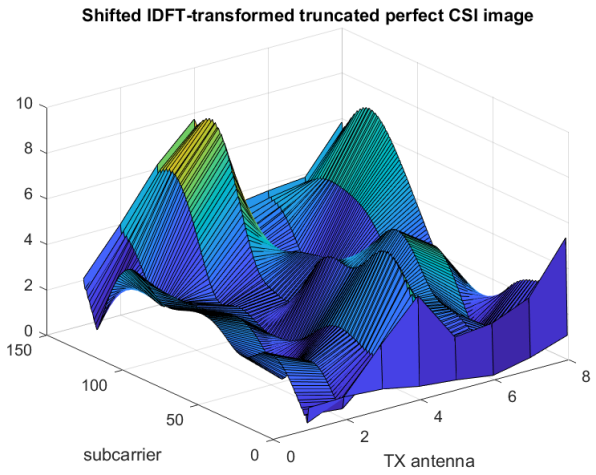


Figure 5.13: Absolute value of truncated $\tilde{H}_{id,CSI}^{IDFT,i}$ matrix, with $K_{DFT} = 4$

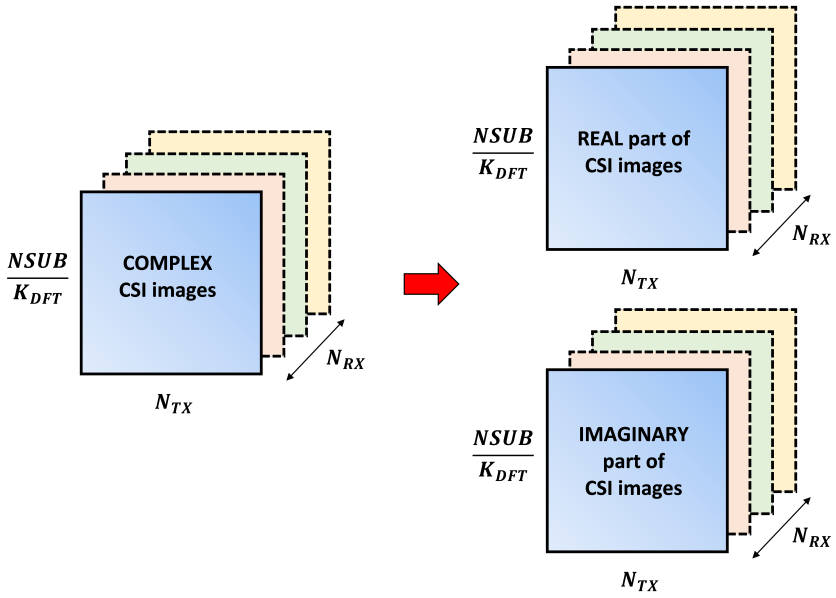


Figure 5.14: Unfolding scheme of CSI real and imaginary parts

Putting together the matrices obtained for each antenna at the receiver, we get the final complex matrix, denoted as $\tilde{H}_{id,CSI}^{IDFT}$. But this is not the matrix we feed to the CNN: a further preprocessing step has to be accomplished, to manage the complex nature of the data we have obtained so far. Also in this case, we follow the approach proposed in reference [23], where the real and the imaginary part of the complex CSI image are treated as different colors and are fed to the network through two independent

input channels. Therefore, each of these N_{RX} input matrices $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}^{\boldsymbol{IDFT,i}}$ must be unfolded into two distinct channels, one for its real part and one for its imaginary part. The unfolding process is schematized in `Figure 5.14`. Then, the matrices obtained have to be normalized, so that all the values at the input of the network range between 0 and 1.

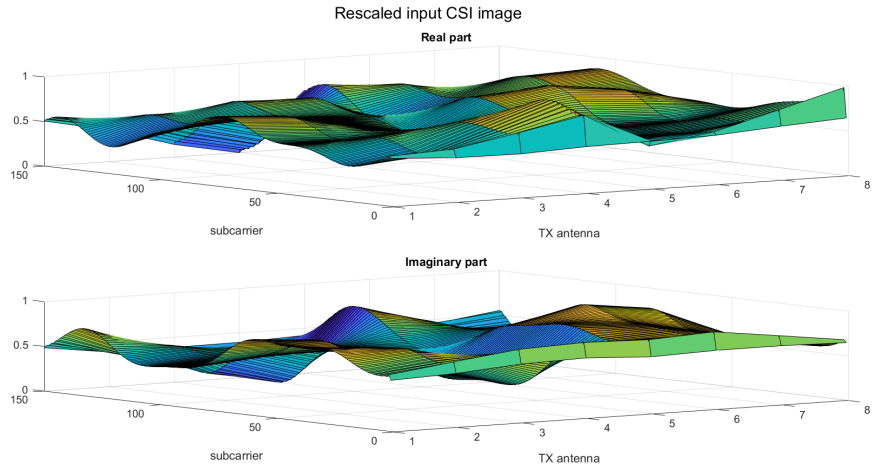


Figure 5.15: Rescaled $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}$ matrix, with $K_{DFT} = 4$

At the end, our dataset consists of $NSUB/K_{DFT} \times N_{TX} \times 2N_{RX}$ matrices, denoted as $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}^{\boldsymbol{IDFT,real}}$. `Figure 5.15` reports the result of the normalization process, considering a single receiver antenna. Remembering the two different multi-receiver approaches (single-channel and multi-channel), illustrated in `Subsection 5.5.2`, `Figure 5.16` summarizes the way data are fed to the network according to the solution adopted:

- when the multi-channel approach is chosen, the matrices $\tilde{\boldsymbol{H}}_{\boldsymbol{id,CSI}}^{\boldsymbol{IDFT,real}}$ are fed to the CNN without any further processing, so that the model presents $2N_{RX}$ input channels (multi-channel model);
- when the single-channel mode is adopted, instead, the real parts of the CSI images, even if related to different receiver antennas, are conveyed to the network on a single input channel, and the same goes for the imaginary parts. In the end, the model presents two input channels, one for the real and one for the imaginary parts of the images (2-channel model).

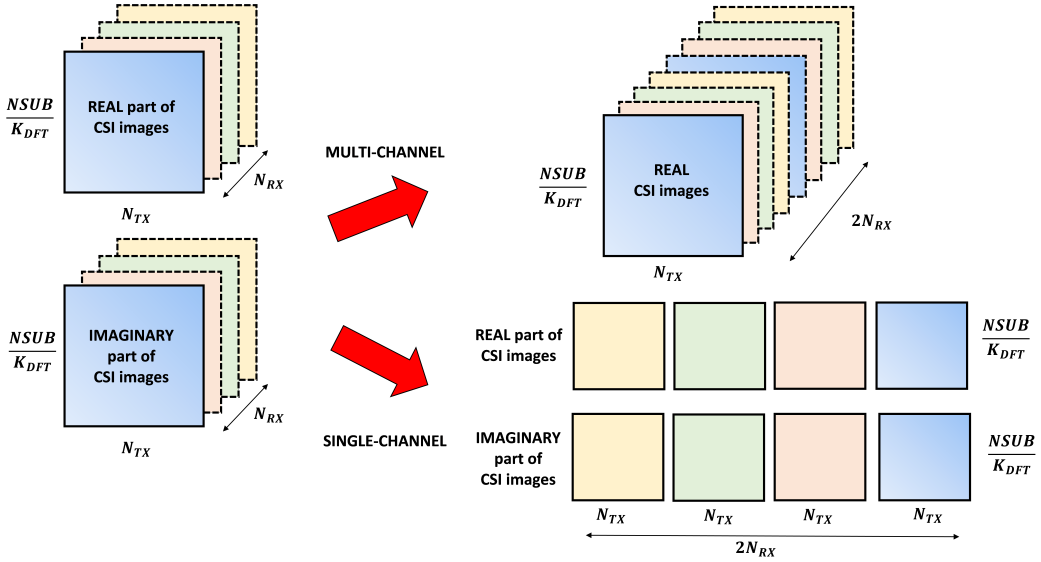


Figure 5.16: Single-channel and multi-channel NR-CsiNet: input data organization

After having presented the data preprocessing steps, `Figure 5.17` and `Figure 5.18` show the complete encoder and decoder structures respectively: as anticipated, the network architecture is substantially inherited from the model presented in paper [23]. The model in the figure assumes the adoption of the multi-channel approach, as is can be deduced from the input layer dimensions of the encoder; in case of single-channel mode, every occurrence of the value $2N_{RX}$ would be replaced by a factor 2.

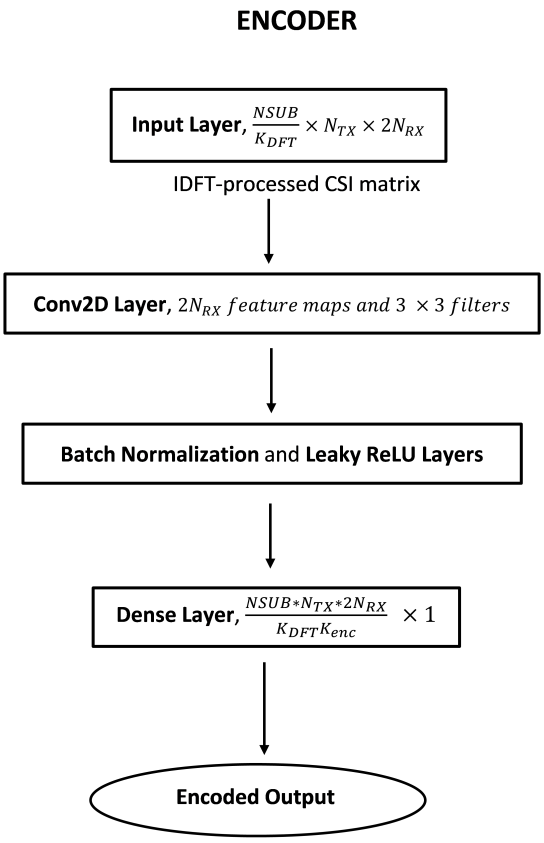


Figure 5.17: NR-CsiNet encoder scheme

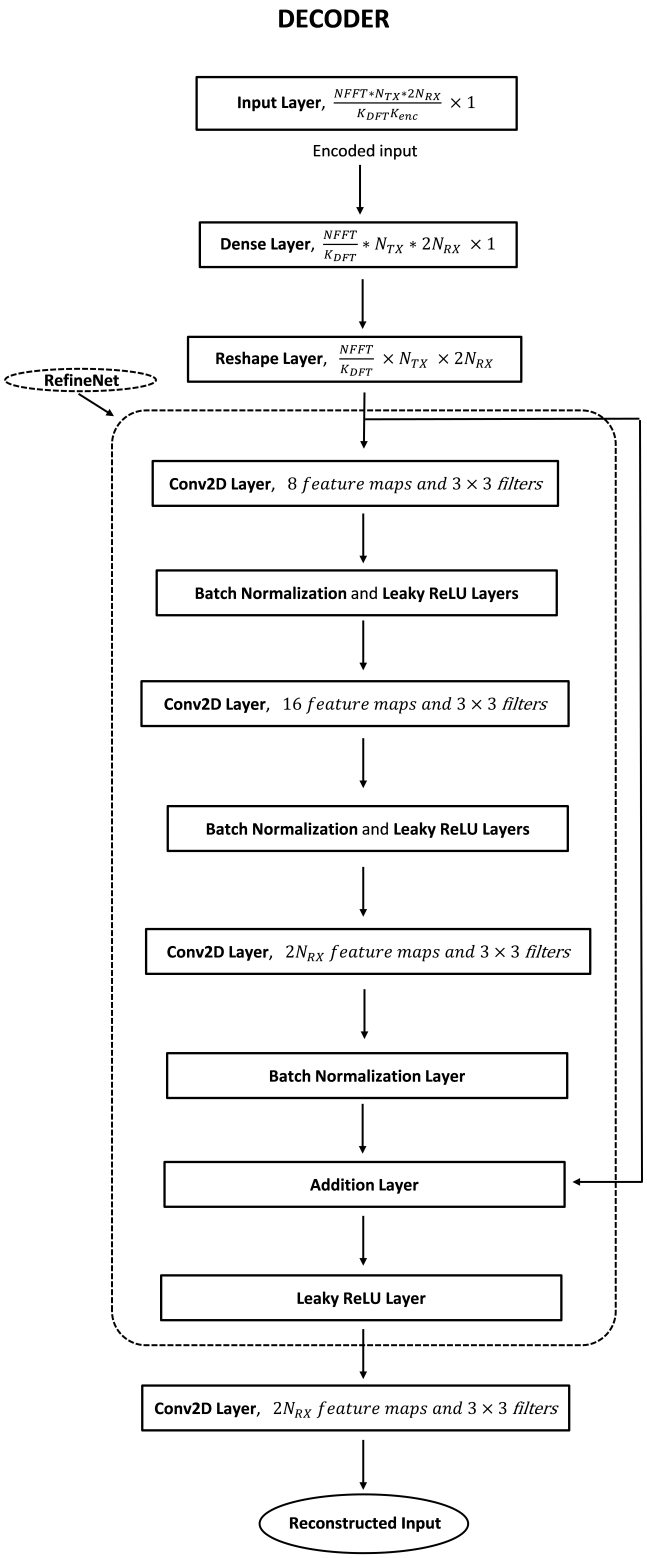


Figure 5.18: NR-CsiNet decoder scheme

5.5.3 NR-CsiNet for CSI feedback estimation and reporting

Instead of focusing only on the compression step, we implement a second version of NR-CsiNet that is able to deal with the challenging downlink channel estimation task: in fact, the input of this second model is no longer the clean CSI image generated by the CDL channel model, as it is assumed in Subsection 5.5.2, but a noisy estimation of it, coming from CSI-RS compensation and interpolation processes. With this in mind, what this model is expected to do is to learn efficient encoding and decoding algorithms that, besides compressing and decompressing the input CSI matrices, must be able to remove the noisy contributions and reconstruct the clean CSI images with the minimum error.

Certainly, this second approach has much more practical validity with respect to the one presented in the previous subsection: in fact, since the assumption of having the perfect CSI information at our disposal is absolutely not realistic, a valuable feedback compression algorithm should not stand apart from the CSI estimation task. However, our model is not entirely realistic: in fact, we consider the noise affecting the CSI-RS transmission in downlink but not the impairments related to the uplink wireless channel, which come into play in the reporting step. In other words, the compressed CSI information, elaborated by the UE starting from a noisy estimate of the CSI-RS pilots, reaches the network side as it is, without any distortion due to the uplink channel. It is therefore unequivocal that the results reported in the following section may not be fully reliable if we consider real scenarios, nevertheless we have made a significant step towards more realistic assumptions.

Data preprocessing

Considering also the CSI estimation step, we have to generate the NR-CsiNet input data exploiting the CSI-RS information: CSI-RS are sent in downlink to the UE so that the downlink channel attenuation can be estimated in correspondence of the known pilot symbols, through a compensation process. We denote as $\boldsymbol{p_{CSI-RS}}$ those $N_{RX} \times N_{TX} \times NSUB$ complex matrices that contain the compensated values in correspondence of the CSI-RS pilots and zeros in all the remaining positions.

Figure 5.19 shows the example of a vector $\boldsymbol{p}^{i,j}_{\boldsymbol{CSI-RS}}$, where i and j represent the transmitter and receiver antenna indexes respectively. The plot reports the absolute values of the complex matrix computed over $NFFT = 1024$ subcarriers: the left and right guard bands occupy 212 subcarriers each, so that only the central $NSUB = 600$ subcarriers are allocated for data transmission and, thus, taken into account. In each of the 50 RBs, composed by 12 subcarriers, only the first two subcarriers host CSI-RS symbols, for a total of 100 CSI-RS REs for each transmitter/receiver antenna

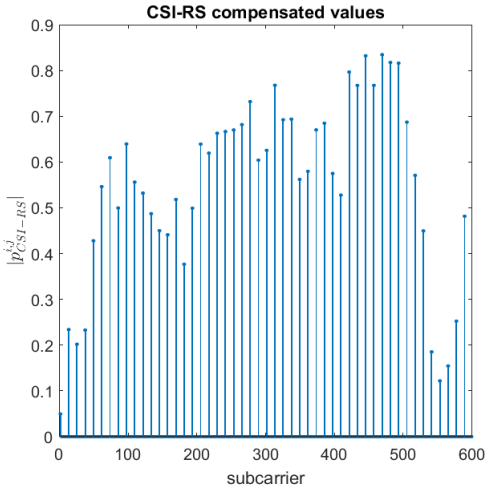


Figure 5.19: $|p^{i,j}_{CSI-RS}|$ vector computed at SINR = 20 dB

pair. To obtain a complete estimation of the downlink CSI matrices $p^{i,j}_{CSI-RS}$ for each transmitter/ receiver antenna pair (i, j), it is necessary to perform an interpolation: in particular, for each $p^{i,j}_{CSI-RS}$ matrix, we perform a cubic RBF interpolation on the real and the imaginary parts separately, using a `MATLAB` predefined library. The result obtained is shown in`Figure 5.20`. We denote the interpolated versions of the estimated CSI images as H_{CSI-RS}.

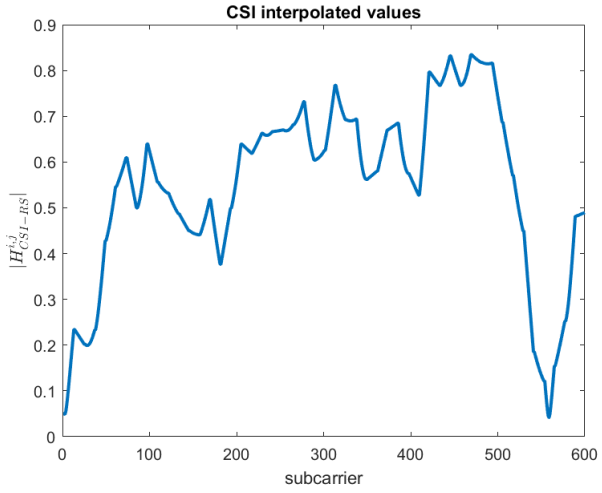


Figure 5.20: $|H^{i,j}_{CSI-RS}|$ vector computed at SINR = 20 dB

At this point, we repeat all the steps described in the `Subsection 5.5.2` for each matrix H^{i}_{CSI-RS}, where i represents the receiver antenna index:

- a DFT-based processing, to sparsify the matrix in the angular-delay domain, obtaining the matrix $\boldsymbol{H}_{\boldsymbol{CSI-RS}}^{\boldsymbol{DFT,i}}$;
- a truncation procedure, to keep only the non-zero rows of matrix $\boldsymbol{H}_{\boldsymbol{CSI-RS}}^{\boldsymbol{DFT,i}}$; the result is an $NSUB/K_{DFT} \times N_{TX}$ matrix $\tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{DFT,i}}$, where K_{DFT} represents the compression factor;
- an Inverse Discrete Fourier Transform (IDFT)-based processing, to bring the $\tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{DFT,i}}$ matrix back to the spatial-frequency domain, obtaining a smoother CSI image $\tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{IDFT,i}}$;
- a min-max normalization process, to scale all the matrix entries to the range [0,1], according to 4.22;
- an unfolding procedure, to split each normalized complex matrix $\tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{IDFT,i}}$ into two real matrices, one for the real and one for the imaginary part.

At the end of these preprocessing steps, we obtain $NSUB/K_{DFT} \times N_{TX} \times 2N_{RX}$ matrices, denoted as $\tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{IDFT,real}}$. Then, before being delivered to the autoencoder, the training data are organized on 2 or $2N_{RX}$ input channels, according to the multi-receiver approach adopted (`Subsection 5.5.2`). In conclusion, the training dataset of our NR-CsiNet is composed by $(\tilde{\boldsymbol{H}}_{\boldsymbol{CSI,id}}^{\boldsymbol{IDFT,real}}, \tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{IDFT,real}})$ pairs, where matrices $\tilde{\boldsymbol{H}}_{\boldsymbol{CSI-RS}}^{\boldsymbol{IDFT,real}}$ represent the clean CSI images of the model, whose generation process is described in `Subsection 5.5.2`.

Variability of channel quality

The $\boldsymbol{p}_{\boldsymbol{CSI-RS}}$ matrices are collected simulating different conditions of the channel: practically, we select a couple of different SINR values, 10 dB and 20 dB, and we train a different NR-CsiNet model for each of these values. Then, we test the behavior of those models at different levels of channel quality: to be precise, simulations are performed with SINR values varying between -5 dB and 35 dB, with a 5 dB granularity.

5.5.4 Variational autoencoders

The NR-CsiNet models described so far are based on the standard autoencoder proposed by the authors of paper [23]. We decide to implement alternative NR-CsiNet versions based on **Variational AutoEncoders** (**VAEs**). VAEs have emerged as one of the most popular techniques for unsupervised learning tasks [6]; their versatility derives from the fact that they can be built on top of efficient function approximators, like neural networks, and the training process can be performed through stochastic gradient

descent algorithm. Moreover, compared to the traditional autoencoders, VAEs allow for significant dimensionality reduction at the cost of a typically negligible loss of accuracy. With these great premises, we try to apply the VAE technique to our CSI feedback compression and estimation tasks. Before moving on to implementation details, it is useful to review the mathematical principle behind VAEs.

Latent Variable Models

Reference [W11] defines VAEs as probabilistic generative models which learn the probability distribution $P(\boldsymbol{x})$ characterizing the input space $\boldsymbol{X}$. The idea is to obtain a good approximation of $P(\boldsymbol{x})$ from which it is possible to generate entries very similar to the input data by means of a simple sampling procedure. In the same reference, VAEs are classified as Latent Variable Model (LVM), referring to the fact that the random vector $\boldsymbol{x} \in \boldsymbol{X}$ of the observed variables can be modeled as a function of a random vector $\boldsymbol{z} \in \boldsymbol{Z}$ of lower dimensionality; the elements of $\boldsymbol{z}$ are defined as unobserved or **latent variables**, since given a datapoint of the input space $\boldsymbol{X}$ generated by the model, the latent varaibles corresponding to it are not necessarily known. Even if it is not a rule, for vector $\boldsymbol{z}$ a Gaussian prior distribution is generally assumed. In reference [6], the model is formalized as follow: given a vector of latent variables $\boldsymbol{z} \in \boldsymbol{Z}$, that can be easily sampled according to the probability density function $P(\boldsymbol{z})$ defined over $\boldsymbol{Z}$, there exists a family of deterministic functions $f(\boldsymbol{z};\boldsymbol{\theta})$ parametrized by the vector $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $f : \boldsymbol{Z} \times \boldsymbol{\Theta} \rightarrow \boldsymbol{X}$. It can be noticed that f is a deterministic function, $\boldsymbol{\theta}$ is fixed but $\boldsymbol{z}$ is random vector, implying that $f(\boldsymbol{z};\boldsymbol{\theta})$ is a random vector in the space $\boldsymbol{X}$. The VAE model aims at optimizing the parameter $\boldsymbol{\theta}$ so that, sampling a vector $\boldsymbol{z}$ from $P(\boldsymbol{z})$, the function $f(\boldsymbol{z};\boldsymbol{\theta})$ returns a datapoint as similar as possible to the entries of the input dataset. This means maximizing the probability that each $\boldsymbol{x}$ generated by the model belongs to the training dataset, which can be expresses as

$$P(\boldsymbol{x}) = \int P(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})P(\boldsymbol{z})\, d\boldsymbol{z} \tag{5.12}$$

It is important to observe that the generic term $f(\boldsymbol{z};\boldsymbol{\theta})$ is replaced with the probability distribution $P(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$, which makes the dependence of $\boldsymbol{X}$ on $\boldsymbol{z}$ explicit. In VAEs, a typical choice for the conditional probability density function $P(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})$ is

$$P(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta}) = N(\boldsymbol{X}|f(\boldsymbol{z};\boldsymbol{\theta}), \sigma^2\boldsymbol{I}) \tag{5.13}$$

where σ is an hyperparameter and $\boldsymbol{I}$ is the identity matrix. The choice of having a Gaussian distribution allows to use the gradient descent optimization algorithm to increase $P(\boldsymbol{X})$ by gradually making $f(\boldsymbol{z};\boldsymbol{\theta})$ more similar to $\boldsymbol{x}$ for some $\boldsymbol{z}$ [6]. Effectively

summarizing the concept of LVMs, reference [W11] defines them as models that use the rules of conditional probability to specify complicated probability distributions over high dimensional spaces, by composing simpler probability density functions.

An overview on VAEs

In paper [6], the authors identify several key issues that VAEs have to deal with: first of all, how to define the latent variables z and which information they represent; secondly, how to define the latent structure behind them, i.e. to define the dependencies between different latent dimensions. Indeed, VAEs adopt a very simple approach to solve these problems: basing on the principle that any d-dimensional distribution can be potentially generated by sampling d variables normally distributed and than mapping these variables through a complicated function, VAEs simply draw independent samples of $\boldsymbol{z}$ from a Gaussian distribution $N(0, \boldsymbol{I})$, avoiding any interpretation of the latent dimensions. This means that, given a powerful function approximator, like a multi-layer neural network, what the VAE model has to do is simply to learn a function which maps the Gaussian independent z variables into latent variables and then map the latent variables to $\boldsymbol{x}$ datapoints. To conclude, reference [6] highlights how, at the end, no prior latent structure is required: the only issue is to determine a suitable expression for $P(\boldsymbol{x})$, so that stochastic gradient descent algorithm can be applied to optimize the model. In practice, since most of $\boldsymbol{z}$ realizations present conditional probability $P(\boldsymbol{x}|\boldsymbol{z})$ close to zero, a new function $Q(\boldsymbol{z}|\boldsymbol{x})$ is defined on the set of $\boldsymbol{z}$ samples which are likely to produce $\boldsymbol{x}$ datapoints, to exclude all the other samples from the computation of $P(\boldsymbol{x})$. Paper [6] motivates this intermediate step by observing that the $\boldsymbol{Z}$ space on which Q is defined is in principle much smaller than the original one, allowing for a relatively easy computation of the conditional expectation $E_{z\sim Q}P(\boldsymbol{x}|\boldsymbol{z})$. In the same paper, the fundamental relationship between $E_{z\sim Q}P(\boldsymbol{x}|\boldsymbol{z})$ and $P(\boldsymbol{x})$ is analyzed in detail, introducing the fundamental concept of **Kullback-Leibler divergence**, that is often referred to with the symbol D.

The Kullback-Leibler divergence between $P(\boldsymbol{z}|\boldsymbol{x})$ and $Q(\boldsymbol{z})$ is expressed as

$$D[Q(\boldsymbol{z})||P(\boldsymbol{z}|\boldsymbol{x})] = E_{z\sim Q}[log(Q(\boldsymbol{z})) - log(P(\boldsymbol{z}|\boldsymbol{x}))] \tag{5.14}$$

By applying the Bayes rule and rearranging the equation, the resulting expression is

$$log(P(\boldsymbol{x})) - D[Q(\boldsymbol{z}|\boldsymbol{x})||P(\boldsymbol{z}|\boldsymbol{x})] = E_{z\sim Q}[log(P(\boldsymbol{x}|\boldsymbol{z}))] - D[Q(\boldsymbol{z}|\boldsymbol{x})||P(\boldsymbol{z})] \tag{5.15}$$

This formula is the core part of VAE models: the left term is the quantity to be maximized, while the right term can be optimized thorugh SGD choosing the

appropriate Q function. In practice, what the models has to do is maximizing the first term $log(P(\boldsymbol{x}))$ and simultaneously minimizing the second quantity $D[Q(\boldsymbol{z}|\boldsymbol{x})||P(\boldsymbol{z}|\boldsymbol{x})]$, which represent an error term. Assuming to use a high-capacity model for $Q(\boldsymbol{z}|\boldsymbol{x})$, $Q(\boldsymbol{z}|\boldsymbol{x})$ actually matches $P(\boldsymbol{z}|\boldsymbol{x})$ so that the error term is nullified and what the model maximizes is indeed the quantity of interest $log(P(\boldsymbol{x}))$.

At this point, the SGD must be applied to the right side of the equation and paper [6] formalizes the procedure. Assuming $Q(\boldsymbol{z}|\boldsymbol{x}) = N(\boldsymbol{z}|\mu(\boldsymbol{x};\boldsymbol{\theta}), \Sigma(\boldsymbol{x};\boldsymbol{\theta}))$, where $\mu(\boldsymbol{x};\boldsymbol{\theta})$ (mean vector) and $\Sigma(\boldsymbol{x};\boldsymbol{\theta})$ (covariance matrix) are deterministic functions parametrized by $\boldsymbol{\theta}$ and learned from data by means of neural networks, and choosing $\boldsymbol{z} \sim N(0, \boldsymbol{I})$, the Kullback-Leibler divergence term $D[Q(\boldsymbol{z}|\boldsymbol{x})||P(\boldsymbol{z})]$ (omitting $\boldsymbol{\theta}$ dependency) becomes

$$D[N(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}))||N(0, \boldsymbol{I})] = \frac{1}{2}[tr(\Sigma(\boldsymbol{x})) + \mu(\boldsymbol{x})\mu(\boldsymbol{x})^T - k + log(det(\Sigma(\boldsymbol{x})))] \quad (5.16)$$

where k represents the dimensionality of the distributions. Some useful observations can be made about this formula:

- since $\Sigma(\boldsymbol{x})$ is constrained to be a diagonal matrix, its determinant consists of the product of all the diagonal entries;
- the term $\mu(\boldsymbol{x})\mu(\boldsymbol{x})^T$ is simplified into a sum over the squared entries of the vector $\mu(\boldsymbol{x})$.

Regarding the term $E_{z\sim Q}[log(P(\boldsymbol{x}|\boldsymbol{z}))]$, the standard SGD procedure is followed: only a single realization of $\boldsymbol{z}$ is considered and $P(\boldsymbol{x}|\boldsymbol{z})$ for that $\boldsymbol{z}$ is taken as approximation of $E_{z\sim Q}[log(P(\boldsymbol{x}|\boldsymbol{z}))]$, avoiding to operate on a huge set of $\boldsymbol{z}$ samples.

Without insisting on mathematical aspects, what is important to underline for implementation purposes is that the Kullback-Leibler divergence term, expressed by (5.16), must be added to the reconstruction loss when implementing a VAE model.

A VAE model for CSI feedback reporting: NR-CsiVAE

In parallel to the NR-CsiNet models described in the previous sections, we develop alternative solutions based on the employment of variational autoencoders, referred to as NR-CsiVAE models. The structure of the CNN remains substantially the same, if not for the inclusion of an **inference network** which approximates the variational parameters $\mu_{\boldsymbol{\phi}}(x)$ and $\sigma_{\boldsymbol{\phi}}(x)$, given the observed variables x. Thus, instead of learning variational parameters for each observed variable x, the idea is to learn a set of K *global* latent parameters $\boldsymbol{\phi}$, which constitute the parameters of the inference network [W14]. In other words, the outputs of the inference network consist of a mean $\mu_{\boldsymbol{\phi}}(\boldsymbol{z})$ and a log-variance $log(\sigma^2_{\boldsymbol{\phi}}(\boldsymbol{z}))$ K-dimensional vectors. `Figure 5.21` reports the scheme of the varational encoder implemented for our NR-CsiVAE model; the dash pane

highlights the portion of the model that differs from the network structure illustrated in Subsection 5.5.2.

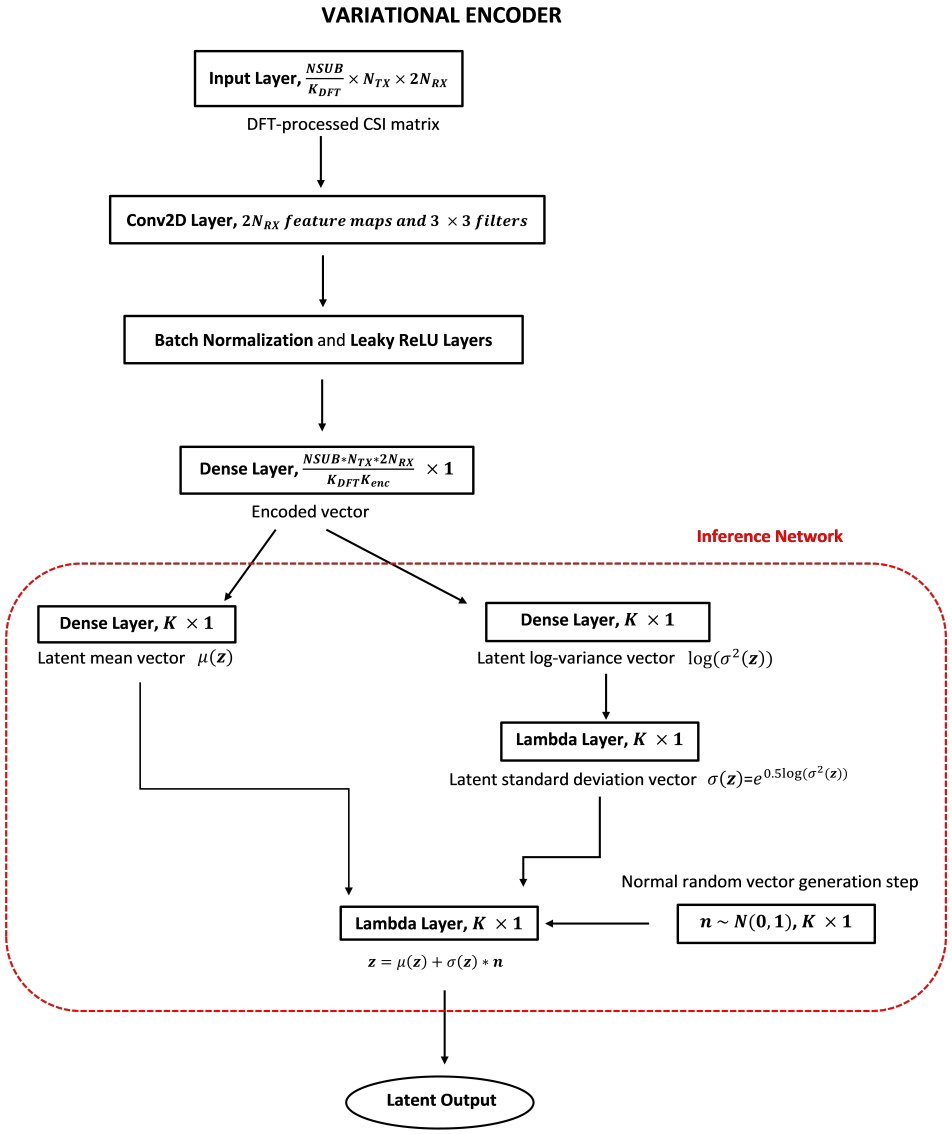


Figure 5.21: Variational encoder scheme for NR-CsiVAE model

Figure 5.22 instead shows the variational decoder structure: it is possible to observe that, compared to the model described in Subsection 5.5.2, a single dense layer is added. We set the number of RefineNet to 2 since, as suggested in paper [23], increasing this number would lead to a significant increase in complexity without a substantial improvement of the performances.

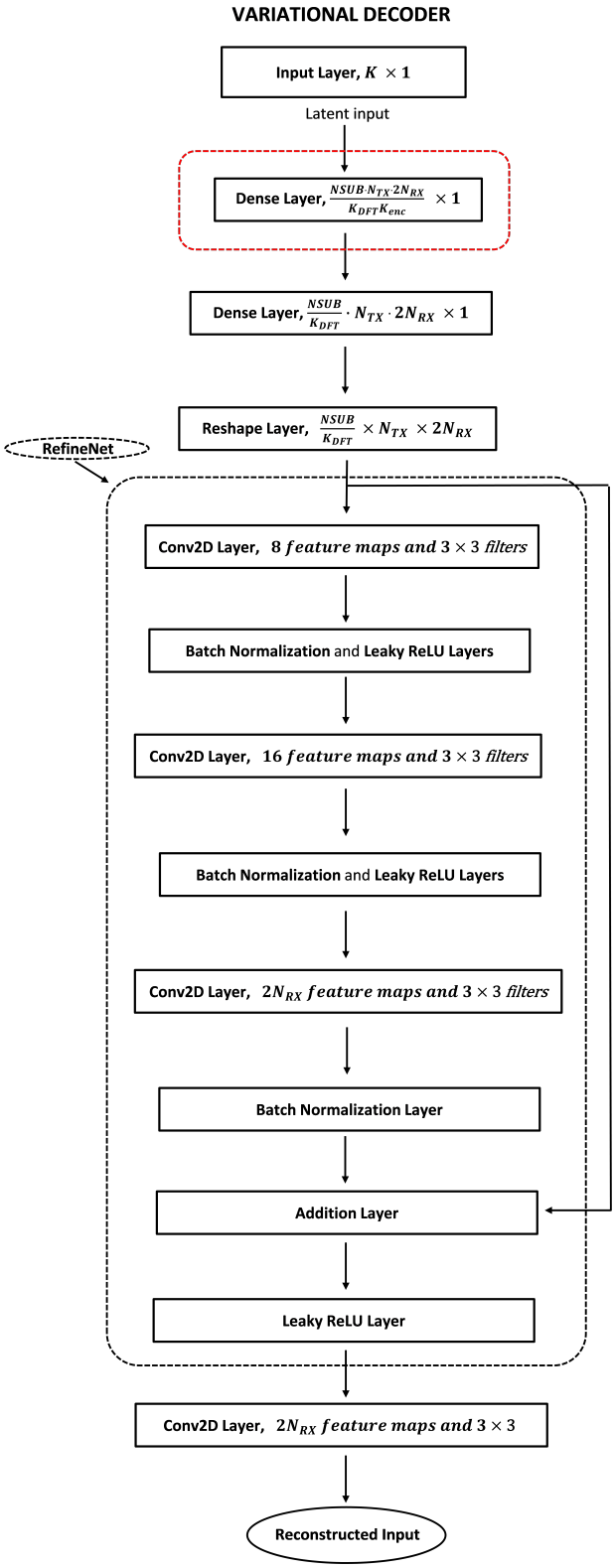


Figure 5.22: Variational decoder scheme for NR-CsiVAE model

As regards the loss computation, we sum to the reconstruction loss the Kullback-Leibler divergence term, derived in 5.16; the relative Python code is reported in Listing 5.1.

Listing 5.1: Python code for VAE loss computation

```
# computing VAE loss

def vae_loss(weight):
    def loss(y_true, y_pred):
        reconstruction_loss = mse(y_true, y_pred)
        reconstruction_loss *= img_total
        KL_loss = K.sum(z_sigma) + K.sum(K.square(z_mean)) \
                  - K.sum(z_log_var) - latent_dim
        KL_loss *= 0.5
        vae_loss = K.mean(reconstruction_loss + weight*KL_loss)
        return vae_loss
    return loss
```

The parameter weight is a multiplication factor whose purpose is to balance the reconstruction loss and the Kullback-Leibler divergence, so that they result to have the same order of magnitude. In this way, the two terms will have more or less the same impact on the optimization process. Instead of fixing the weighting factor to a static value, we decide to adopt one of the most popular approach for KL divergence balancing, called Kullback-Laibler (KL) annealing, which consists in gradually introducing the KL loss term, increasing its weighting factor from 0 to 1 in KL_annealtime epochs, starting from the epoch KL_start. In this way, for the first KL_start epochs, the network can learn how to reconstruct the input data without taking into account the KL cost: in fact, this term generally assumes very large values at the beginning of the training process, so that the network is forced to ignore the reconstruction task, with the risk of running into a local minimum. With this approach, instead, only after a configurable number of epochs the network can start to focus on the KL loss and optimize the latent space distribution. Reference [W9] suggests a KL annealing implementation by means of a callback that checks and eventually updates the weight term at each training epoch; Listing 5.2 reports the relative Python code.

Listing 5.2: Python code for KL annealing

```
class AnnealingCallback(Callback):
    def __init__(self, weight):
        self.weight = weight
    def on_epoch_end (self, epoch, logs={}):
        if epoch > KL_start :
            new_weight = min(K.get_value(self.weight) \
                             + (1./KL_annealtime), 1)
            K.set_value(self.weight, new_weight)
```

5.6 Simulation results

This section summarizes the results obtained by the NR-CsiNet and the NR-CsiVAE models described in the previous sections. Performances are evaluated after simulation campaigns conducted with the NR link level simulator presented in Section 4.4, replacing the blocks for CSI feedback definition and reporting with our deep learning-based models. The beamforming block at the transmitter side takes in input the CSI matrix compressed and sent back by the UE and performs a Singular Value Decomposition (SVD) operation: the eigenvector correspondent to the higher eigenvalue of the matrix is selected as beamforming vector for the next downlink transmission.

Table 5.2 reports the main parameters of interest concerning the simulator settings, while Table 5.4 shows the hyperparameters adopted for the training and testing process of our model.

Table 5.4: Simulation parameters for 8×2 configuration

Parameter	2-channel model	multi-channel model
Training set	1000 sim of 100 TTI	1500 sim of 100 TTI
Validation set	100 sim of 100 TTI	150 sim of 100 TTI
Testing set	100 sim of 100 TTI	100 sim of 100 TTI
Learning rate	0.001	0.001
Loss function	MSE	MSE
Optimizer	Adam	Adam
Epochs	300	300
# RefineNet units	2	2
K_{DFT}	3	3

The DFT-compression factor should not be confused with the final compression factor K, which is computed as

$$K = \frac{NSUB \cdot N_{TX} \cdot N_{RX}}{encoded\_dim} \tag{5.17}$$

where encoded_dim represents the effective dimension of the encoded feedback vector.

5.6.1 NR-CsiNet for feedback compression

In this subsection, we analyze the performance of our NR-CsiNet model assuming to work with the perfect channel matrices at the receiver side. The convolutional autoencder is trained by means of a set of clean CSI images and its only task is to compress and reconstruct the input matrices, without having to deal with any sort of noise. Clearly, this is not a realistic scenario, but anyway it is a good test bed to check the coding and decoding capabilities of our model.

Model performances are evaluated on the basis of the same indicator parameters analyzed in Section 4.6: NMSE, throughput, raw BER, decoded BER and BLER. It is important to note that the NMSE value, with its relative standard deviation, is not reported for each SINR level, differently from all the other parameters evaluated: this sounds reasonable since, for this subset of experiments, the channel noise is completely neglected and thus, independently from the SINR level considered, the ideal CSI matrix remains the same.

MIMO scenario: 8x2 antenna configuration

This subsection presents the simulation results achieved considering an 8×2 antenna system: the following pages show some plots which graphically summarize the data collected in Section B.1. Two different NR-CsiNet variants are considered:

- 2-channel: real and imaginary parts of all the CSI images, independently of the receiver antenna they are relative to, are conveyed on the same two input channels;
- multi-channel: real and imaginary parts of the CSI images relative to different receiver antenna are conveyed to the network on $2N_{RX}$ distinct channels.

For this configuration, a single value of DFT-compression factor is considered, i.e. $K_{DFT} = 3$. In principle, avoiding the DFT/IDFT preprocessing step both in downstream and in upstream, which means setting K_{DFT} to 1, would certainly lead to a gain in simplicity and speed of execution. Moreover, it should be taken into account that the truncation performed after the DFT step is certainly a "lossy" operation: in fact, the side values which are eliminated are very close to zero, but not exactly zero. This means that a portion of CSI information, however small, is already definitely lost during the preprocessing step. Feeding the NR-CsiNet model with the whole CSI information may turn out to be a better solution also in terms of reconstruction accuracy. The choice of fixing the DFT-compression factor to 3 has been dictated by complexity reasons: since a second receiver antenna comes into play, the number of feature maps for the multi-channel model rises; as a natural consequence, the total number of learnable parameters increases proportionally, giving rise to computational complexity and

memory issues. For this reason, we decide to adopt a higher DFT-compression factor to reduce the autoencoder input dimension. Experiments with lower DFT-compression factors are left for future works.

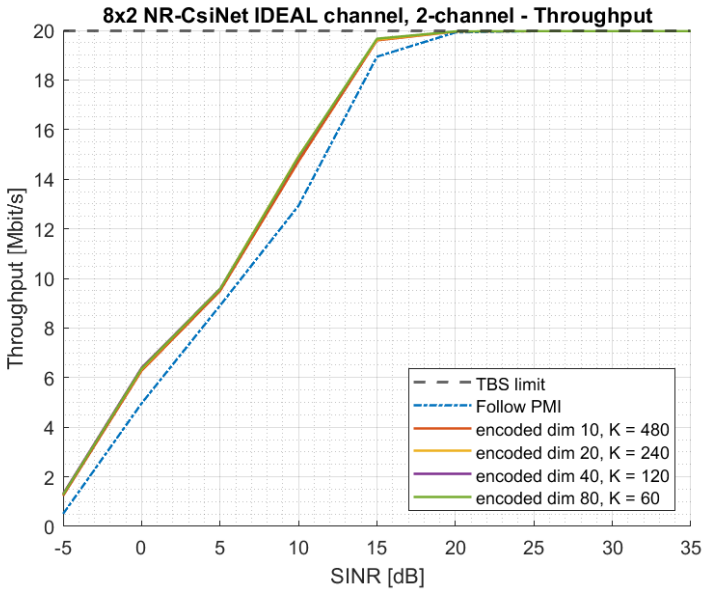
Table 5.5: 8×2 NR-CsiNet **2-channel** model, $K_{DFT} = 3$ -NMSE

encoded dim	**80**	**40**	**20**	**10**
NMSE	0.0017	0.0052	0.0493	0.1682
σ_{NMSE}	0.0005	0.0014	0.0246	0.0634

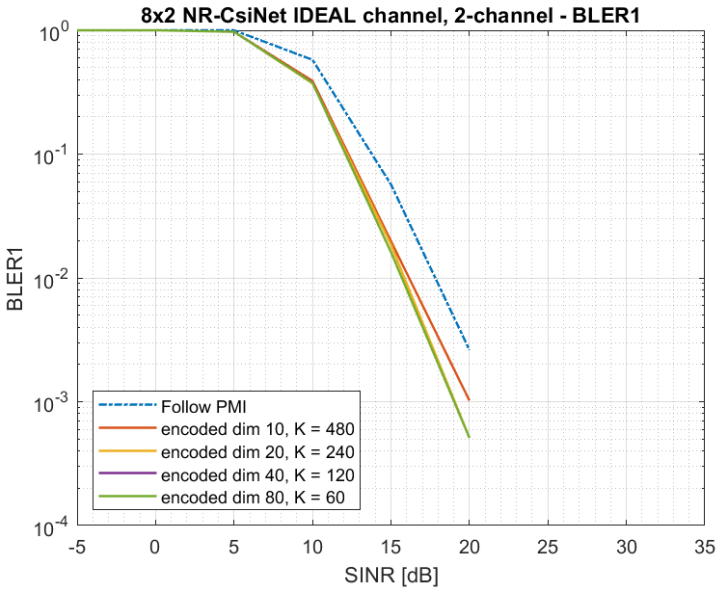
Table 5.6: 8×2 NR-CsiNet **multi-channel** model, $K_{DFT} = 3$ - NMSE

encoded dim	**80**	**40**	**20**	**10**
NMSE	0.0058	0.0440	0.1134	0.2535
σ_{NMSE}	0.0014	0.0135	0.0381	0.0717

`Table 5.5` and `Table 5.6` collect NMSE statistics relative 2-channel model and multi-channel model respectively: the first row contains the mean NMSE values for different compression factor, while the second row reports the corresponding standard deviation.

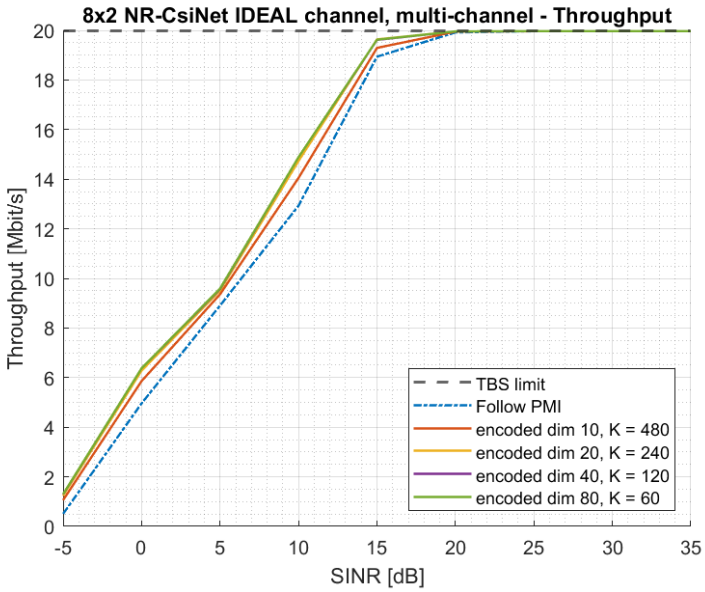


(a) Throughput

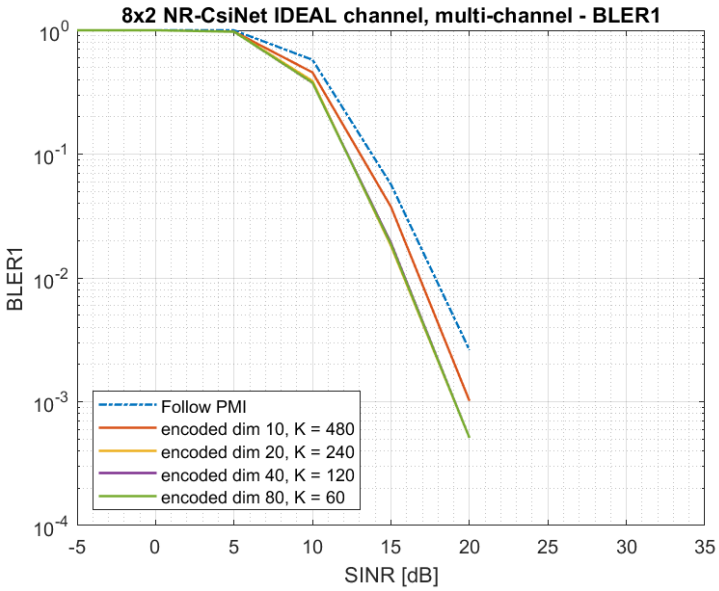


(b) BLER1

Figure 5.23: 8x2 NR-CsiNet IDEAL channel - 2-channel, $K_{DFT} = 3$



(a) Throughput



(b) BLER1

Figure 5.24: 8x2 NR-CsiNet IDEAL channel - multi-channel, $K_{DFT} = 3$

As it can be observed in `Figure 5.23` and in `Figure 5.24`, NR-CsiNet solutions demonstrate excellent performance, even better than the one achieved by Follow PMI. Looking at the plots relative to the different models considered (2-channel and multi-channel), it would seem reasonable to conclude that the two variants achieve comparable results. In fact, although the 2-channel model presents lower NMSE values, as it can be deduced comparing `Table 5.5` and `Table 5.6`, this NMSE disparity does not translate into a substantial performance gap.

Anyway, even if the variants analyzed would appear to be equivalent, there exists a fundamental difference which makes the 2-channel NR-CsiNet model preferable in general. As already explained in `Subsection 4.6.2`, the 2-channel approach is more scalable with the number of antennas at the receiver side, since the number of learnable parameters does not increase proportionally to N_{RX}.

5.6.2 NR-CsiNet for feedback estimation and reporting

The NR-CsiNet models analyzed in this subsection are designed to deal with both the estimation and the reporting tasks: differently from the models evaluated in the previous subsection, these convolutional autoencoders are not fed with clean CSI images but with their noisy estimates, as it is clearly explained in `Subsection 5.5.3`. In view of the observations made in the previous section, these experiments evaluate only the NR-CsiNet 2-channel models; in particular, two different NR-CsiNet solutions

have been trained and tested:

- a NR-CsiNet model trained with data collected at $SINR = 10$ dB;
- a NR-CsiNet model trained with data collected at $SINR = 20$ dB.

MIMO scenario: 8x2 antenna configuration

The figures below show the simulation results obtained in case of 8×2 antenna configuration. Also for these simulations a single value of DFT-compression factor is considered ($K_{DFT} = 3$), again for reasons of memory and computational complexity. As already mentioned, the SINR values selected for the training process are only two, i.e. 10 dB and 20 dB. It is worth noting that performances in terms of NMSE, apart from the case in which K is increased to 480, are almost independent of the SINR level at which training data are collected. The low sensitivity to the SINR value configured for the training process supports our choice to reduce the number of SINR levels considered: instead of collecting training data at five different SINR values, as for NR-ChannelNet model, we limit our experiments to only two SINR alternatives.

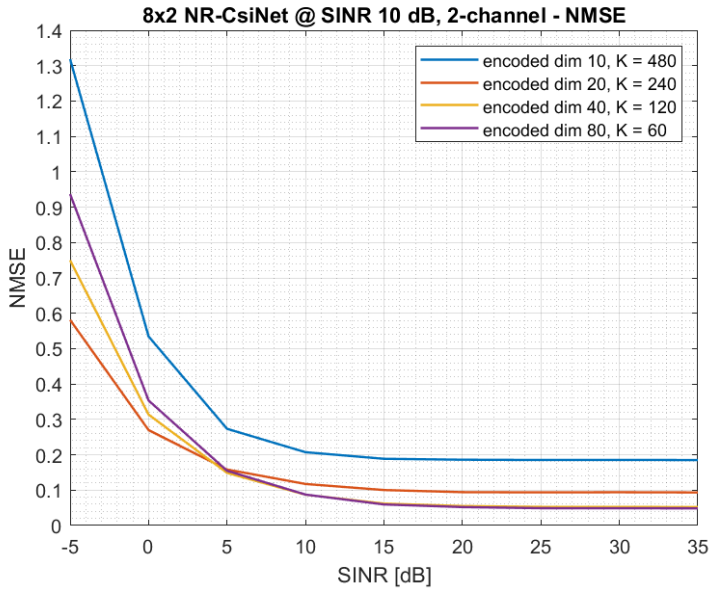


Figure 5.25: 8x2 NR-CsiNet 2-channel @ SINR 10 dB, $K_{DFT} = 3$ - NMSE

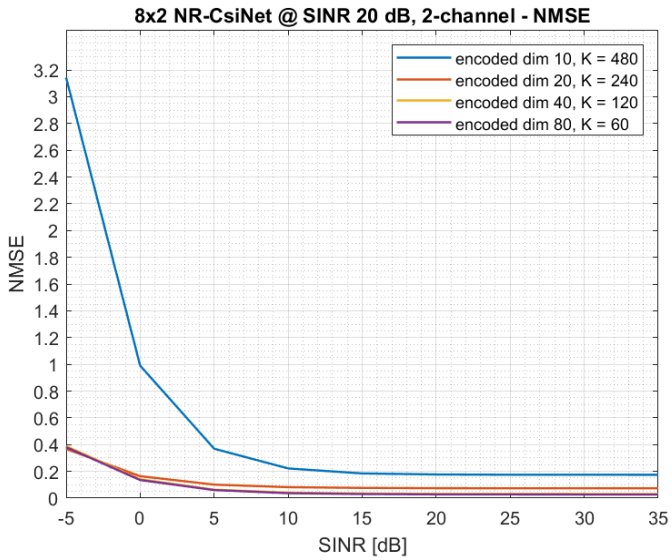
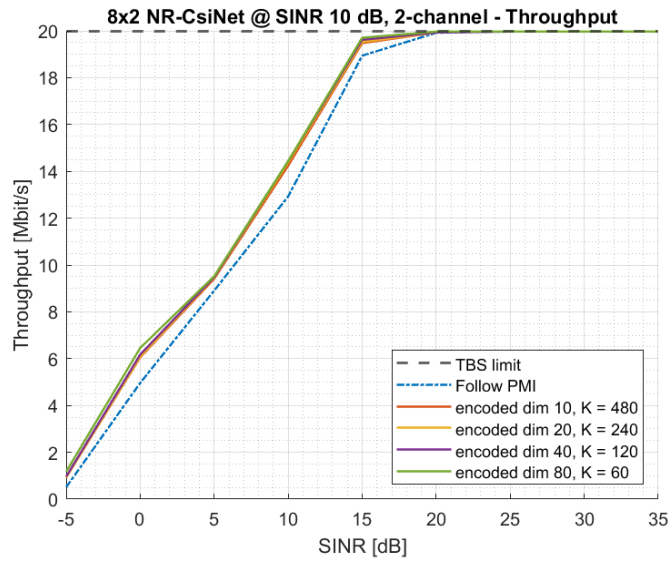
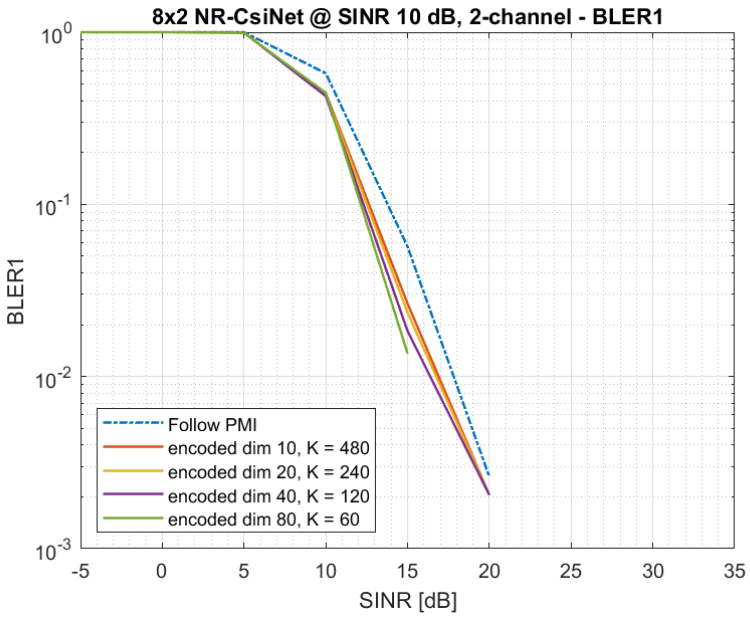


Figure 5.26: 8x2 NR-CsiNet 2-channel @ SINR 20 dB, $K_{DFT} = 3$ - NMSE

Observing the graphs in `Figure 5.27` and `Figure 5.28`, relative to throughput and BLER, it seems reasonable to conclude that, in 8×2 scenarios, our convolutional autoencoder outperforms the Follow PMI, even when dealing with noisy CSI estimates instead of perfect CSI matrices. In particular, the plots show that above 20 dB of SINR, both NR-CsiNet and Follow PMI throughput curves reach the maximum limit imposed by the TBS; at lower SINR, instead, all the variants of our deep learning solution prove better performances.

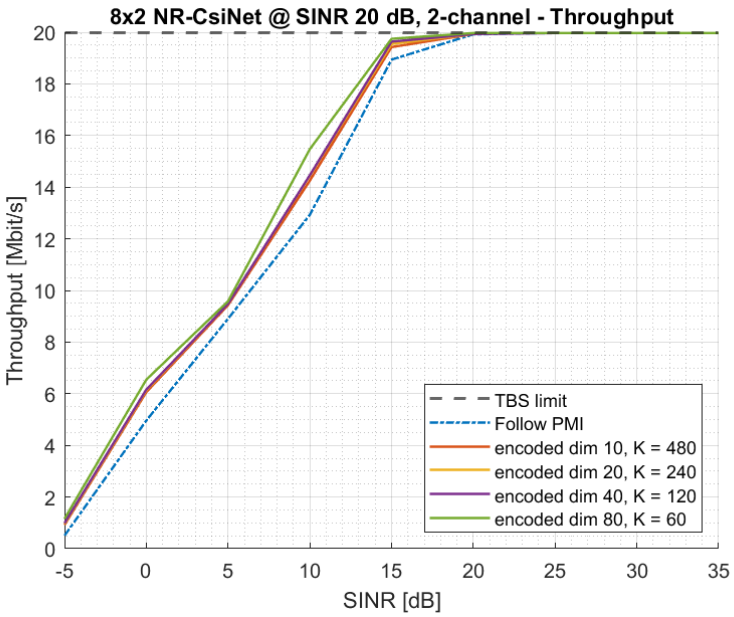


(a) Throughput

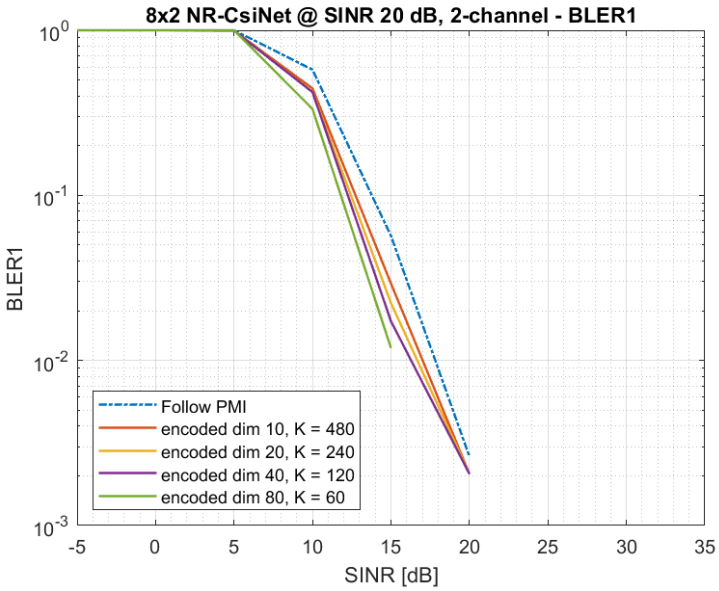


(b) BLER1

Figure 5.27: 8x2 NR-CsiNet @ SINR 10 dB - 2-channel, $K_{DFT} = 3$



(a) Throughput



(b) BLER1

Figure 5.28: 8x2 NR-CsiNet @ SINR 20 dB - 2-channel, $K_{DFT} = 3$

In the end, it is worth noting that for all our experiments we adopted a single variant of channel model, i.e. the Clustered Delay Line (CDL) (`Section 4.4`): simulation results have shown that this type of channel model, although designed for multipath contexts, usually presents a dominant cluster, which implicitly defines a preferential direction for signal transmission. It is clear that, considering such a channel model, the Follow PMI is expected to achieve good performance: in fact, among a set of possible beams, the PMI index simply identifies the one pointing to the receiver with more precision. Our guess is that, considering "richer" channel realizations, where multipath is not dominated by a single cluster of paths, our deep learning-based approach has all the potential to further gain ground on the Follow PMI algorithm.

5.6.3 NR-CsiVAE for feedback estimation and reporting

This last subsection summarizes the results obtained by replacing the standard convolutional autoencoder with the variational autoencoder model (NR-CsiVAE), described in `Subsection 5.5.4`. Also in this case, we limit our experiments to the 8×2 scenario, considering only the 2-channel variant and a single SINR value for the training, i.e. 10 dB. `Table 5.7` shows some specific hyperparameters adopted for VAE training. K_{enc} represents the compression factor computed at the output of the first dense layer of the variational encoder (`Figure 5.21`); it must not be confused with the final compression factor K, evaluated after the data projection into the latent space.

Fixed K_{enc}, two different values of K are examined: 240 and 480.

Table 5.7: Training hyperparamters for NR-CsiVAE

Hyperparameter	Value
Epochs	300
KL_start	150
KL_annealtime	10^6
K_{enc}	4

As already mentioned, compared to the traditional autoencoders, VAEs should allow for significant dimensionality reduction at the cost of a typically negligible loss of accuracy. However, these considerations are especially based on results achieved in image processing applications: moving this technology to a different context of use makes the previous observations less reliable. Right from the training phase, in fact, VAE has proved difficult to adapt to our use case: the gap between reconstruction loss and KL divergence term turned out to be unfeasibly large, making it difficult to simultaneously minimize both quantities. The plots shown below confirm our suspicion: for our case study, the introduction of a VAE does not bring any substantial improvement with respect to the use of a traditional autoencoder; on the contrary, VAE results to be worse-performing, also compared to the Follow PMI algorithm.

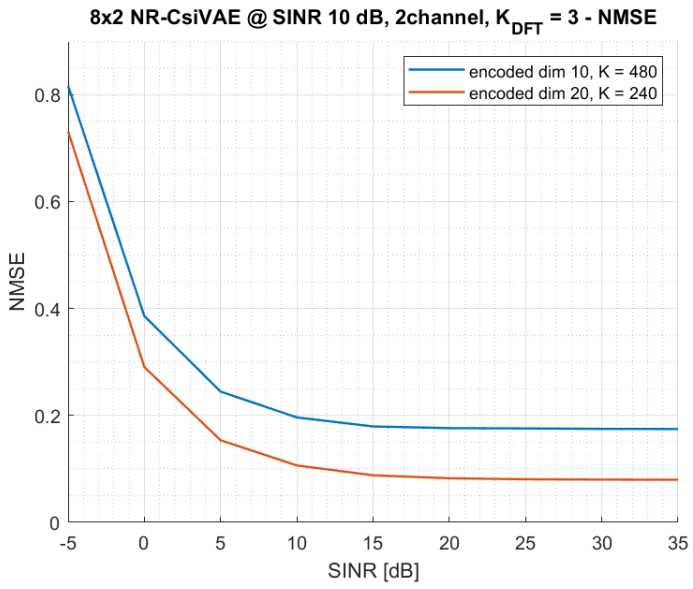
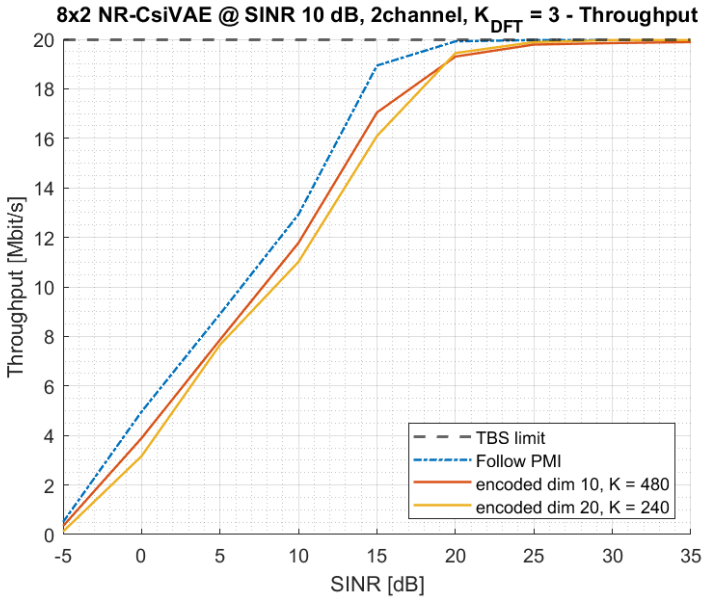
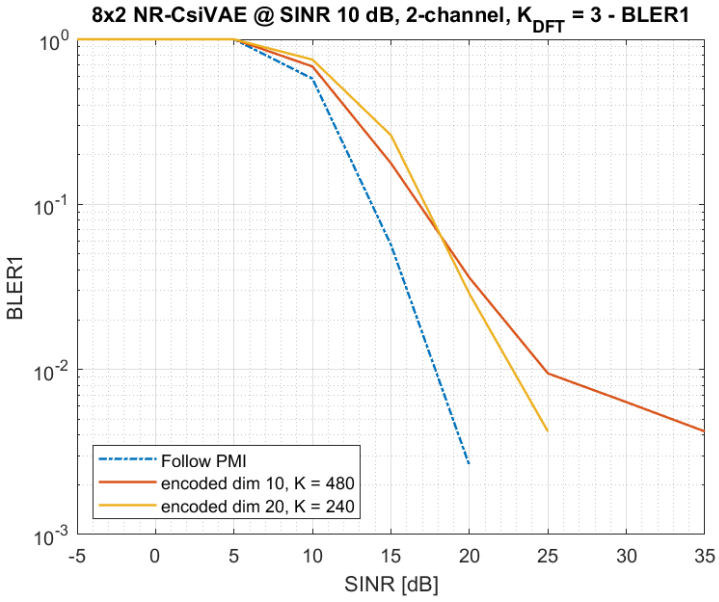


Figure 5.29: 8x2 NR-CsiVAE 2-channel model @ SINR 10 dB, $K_{DFT} = 3$ - NMSE



(a) Throughput



(b) BLER1

Figure 5.30: 8x2 NR-CsiVAE 2-channel model @ SINR 10 dB, $K_{DFT} = 3$

Conclusions

This thesis was born as a descriptive document relative to the research project carried out in collaboration with Telecom Italia. The purpose of this activity was to investigate the potential use of neural networks as an alternative to traditional physical layer blocks, taking into account the peculiar characteristics of 5G-based communication systems.

The first phase was essentially dedicated to activities of research and collection of bibliographical material, with the aim of identifying the most promising applications and finding some previous work to be used as a starting point. On the basis of this preliminary research, two different case studies were selected: the channel estimation and the feedback reporting modules. For both these physical blocks, a deep learning solution was implemented: specific deep neural networks were built and integrated in the New Radio link simulator developed in Telecom Italia laboratories, as replacement of standard algorithms.

Once the implementation task was completed, a final phase of simulation and performance evaluation was started. According to the results obtained, it is not possible to state that the introduction of deep learning mechanisms in the physical transmission chain of 5G systems would lead to substantial improvements. However, going beyond simulation contexts, where reality is simplified and approximated by mathematical models, deep learning solutions have a good chance of gaining ground on standard algorithms, which typically exploit some knowledge of the underlying model.

Nevertheless, this thesis studies the argument with a more exploratory intent: its purpose is to test the waters, providing some useful guide lines for future investments into innovative machine learning-based technologies. The experiments performed and described in these pages clearly demonstrate that deep learning integration into the New Radio transmission chain is not utopian at all, but a concrete possibility.

Appendix A

NR-ChannelNet Simulation Results

A.1 MISO scenario: 8 X 1 antenna configuration

Table A.1: ChannelNet **1-channel** model @ SINR 0 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.051	0.454	1	1	0.978	0.975	1.316
0	0.698	0.419	1	0.883	0.665	0.760	0.103
5	1.581	0.356	0.982	0.454	0.464	0.170	0.048
10	2.707	0.241	0.615	0.213	0.107	0	0.073
15	3.580	0.134	0.323	0.120	0.006	0	0.093
20	4.178	0.061	0.133	0.023	0.01	0	0.098
25	4.451	0.033	0.069	0	0	0	0.070
30	4.484	0.028	0.059	0	0	0	0.056
35	4.484	0.028	0.059	0	0	0	0.055

Table A.2: ChannelNet **2-channel** model @ SINR 0 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.051	0.455	1	1	0.978	0.975	1.008
0	0.680	0.419	1	0.901	0.649	0.774	0.404
5	1.568	0.358	0.979	0.453	0.480	0.208	0.089
10	2.625	0.250	0.648	0.222	0.121	0.005	0.068
15	3.542	0.136	0.345	0.097	0	0	0.053
20	4.079	0.071	0.162	0.036	0	0	0.062
25	4.326	0.047	0.099	0	0	0	0.058
30	4.367	0.042	0.088	0	0	0	0.061
35	4.341	0.044	0.095	0	0	0	0.061

Table A.3: ChannelNet **1-channel** model @ SINR 10 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.056	0.456	1	1	0.978	0.970	0.102
0	0.713	0.419	1	0.913	0.630	0.675	0.102
5	1.571	0.360	0.993	0.450	0.462	0.174	0.203
10	2.768	0.238	0.602	0.216	0.107	0	0.106
15	3.659	0.122	0.293	0.107	0	0	0.094
20	4.382	0.039	0.086	0	0	0	0.096
25	4.555	0.023	0.042	0	0	0	0.095
30	4.604	0.017	0.031	0	0	0	0.095
35	4.614	0.016	0.029	0	0	0	0.096

Table A.4: ChannelNet **2-channel** model @ SINR 10 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.066	0.455	1	1	0.978	0.960	0.383
0	0.665	0.422	1	0.922	0.660	0.712	0.246
5	1.515	0.361	0.991	0.471	0.498	0.197	0.064
10	2.633	0.247	0.636	0.239	0.130	0.007	0.067
15	3.575	0.133	0.324	0.118	0.025	0	0.059
20	4.275	0.049	0.110	0.020	0	0	0.045
25	4.522	0.026	0.050	0	0	0	0.053
30	4.548	0.024	0.044	0	0	0	0.059
35	4.583	0.020	0.036	0	0	0	0.058

Table A.5: ChannelNet **1-channel** model @ SINR 20 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.061	0.457	1	1	0.978	0.965	0.110
0	0.700	0.419	1	0.909	0.634	0.713	0.102
5	1.591	0.360	0.993	0.455	0.444	0.138	0.197
10	2.724	0.241	0.620	0.217	0.102	0	0.098
15	3.672	0.120	0.288	0.103	0	0	0.102
20	4.385	0.039	0.086	0	0	0	0.094
25	4.588	0.019	0.035	0	0	0	0.096
30	4.619	0.015	0.027	0	0	0	0.095
35	4.629	0.014	0.025	0	0	0	0.095

Table A.6: ChannelNet **2-channel** model @ SINR 20 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.053	0.456	1	1	0.978	0.973	0.140
0	0.667	0.420	1	0.910	0.660	0.737	0.075
5	1.505	0.362	0.993	0.468	0.527	0.215	0.145
10	2.686	0.242	0.625	0.233	0.0931	0	0.087
15	3.641	0.126	0.295	0.125	0.0250	0	0.084
20	4.352	0.041	0.091	0.011	0	0	0.096
25	4.558	0.023	0.042	0	0	0	0.095
30	4.601	0.018	0.032	0	0	0	0.094
35	4.624	0.015	0.026	0	0	0	0.095

Table A.7: ChannelNet **1-channel** model @ SINR 50 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.061	0.457	1	1	0.978	0.965	0.141
0	0.642	0.422	1	0.936	0.645	0.731	0.054
5	1.479	0.365	0.996	0.485	0.519	0.273	0.141
10	2.559	0.255	0.671	0.240	0.131	0	0.091
15	3.560	0.138	0.330	0.116	0.025	0	0.083
20	4.329	0.043	0.098	0.008	0	0	0.095
25	4.588	0.019	0.034	0	0	0	0.095
30	4.621	0.015	0.027	0	0	0	0.095
35	4.637	0.013	0.023	0	0	0	0.095

Table A.8: ChannelNet **2-channel** model @ SINR 50 dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.038	0.455	1	1	0.978	0.988	0.153
0	0.644	0.422	1	0.917	0.665	0.731	0.082
5	1.479	0.366	0.996	0.500	0.510	0.228	0.145
10	2.554	0.254	0.670	0.238	0.137	0	0.087
15	3.585	0.135	0.321	0.108	0.025	0	0.084
20	4.349	0.042	0.091	0.014	0	0	0.096
25	4.558	0.023	0.042	0	0	0	0.094
30	4.611	0.016	0.029	0	0	0	0.095
35	4.621	0.015	0.027	0	0	0	0.095

Table A.9: ChannelNet **1-channel** model @ SINR [0,20] dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.069	0.453	1	1	0.978	0.958	1.445
0	0.693	0.420	1	0.902	0.632	0.735	0.221
5	1.520	0.360	0.991	0.459	0.521	0.200	0.074
10	2.686	0.241	0.615	0.242	0.116	0.014	0.061
15	3.682	0.121	0.285	0.098	0.025	0	0.092
20	4.377	0.040	0.087	0.003	0	0	0.085
25	4.596	0.018	0.033	0	0	0	0.067
30	4.616	0.016	0.028	0	0	0	0.057
35	4.619	0.015	0.027	0	0	0	0.054

Table A.10: ChannelNet **2-channel** model @ SINR [0,20] dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.031	0.459	1	1	0.978	0.995	2.813
0	0.680	0.418	1	0.898	0.636	0.772	0.944
5	1.591	0.360	0.991	0.468	0.427	0.139	0.086
10	2.674	0.244	0.631	0.217	0.116	0	0.079
15	3.633	0.126	0.307	0.108	0	0	0.069
20	4.346	0.044	0.094	0.008	0	0	0.046
25	4.558	0.023	0.042	0	0	0	0.054
30	4.609	0.017	0.030	0	0	0	0.070
35	4.621	0.015	0.027	0	0	0	0.056

Table A.11: ChannelNet **1-channel** model @ SINR [20,50] dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.051	0.454	1	1	0.978	0.975	2.047
0	0.662	0.422	1	0.911	0.653	0.776	0.285
5	1.538	0.360	0.992	0.455	0.481	0.227	0.087
10	2.671	0.244	0.623	0.227	0.143	0.029	0.063
15	3.669	0.124	0.292	0.102	0.025	0	0.087
20	4.390	0.040	0.084	0.003	0	0	0.088
25	4.601	0.018	0.031	0	0	0	0.068
30	4.627	0.014	0.025	0	0	0	0.055
35	4.634	0.013	0.024	0	0	0	0.054

Table A.12: ChannelNet **2-channel** model @ SINR [20,50] dB - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.056	0.454	1	1	0.978	0.970	2.033
0	0.654	0.421	1	0.907	0.656	0.784	0.264
5	1.543	0.359	0.992	0.457	0.465	0.205	0.084
10	2.699	0.242	0.609	0.228	0.143	0.017	0.064
15	3.633	0.128	0.308	0.098	0.025	0	0.086
20	4.359	0.043	0.091	0.003	0	0	0.088
25	4.563	0.022	0.040	0	0	0	0.068
30	4.621	0.015	0.027	0	0	0	0.056
35	4.637	0.013	0.023	0	0	0	0.054

Table A.13: 2D-MMSE standard algorithm - 8x1 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.055	0.456	1	1	0.950	0.950	2.874
0	0.655	0.419	1	0.927	0.680	0.465	1.943
5	1.327	0.367	0.964	0.634	0.500	0.220	0.182
10	2.313	0.270	0.770	0.331	0.116	0.010	0.126
15	3.458	0.143	0.388	0.108	0.014	0	0.132
20	4.158	0.069	0.162	0.009	0	0	0.142
25	4.406	0.041	0.088	0	0	0	0.150
30	4.483	0.031	0.064	0	0	0	0.116
35	4.505	0.028	0.058	0	0	0	0.118

A.2 MIMO scenario: 32 X 2 antenna configuration

Table A.14: ChannelNet **2-channel** model @ SINR 0 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.250	0.393	1	0.740	0.513	0.185	0.138
0	2.098	0.315	0.957	0.188	0.056	0	0.130
5	3.277	0.164	0.458	0.006	0	0	0.099
10	4.234	0.055	0.125	0	0	0	0.085
15	4.624	0.013	0.025	0	0	0	0.087
20	4.698	0.004	0.008	0	0	0	0.084
25	4.672	0.007	0.016	0	0	0	0.105
30	4.677	0.007	0.014	0	0	0	0.083
35	4.675	0.007	0.015	0	0	0	0.083

Table A.15: ChannelNet **multi-channel** model @ SINR 0 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.433	0.414	1	0.960	0.797	0.782	0.614
0	1.240	0.370	0.995	0.656	0.495	0.410	0.227
5	1.833	0.312	0.890	0.367	0.316	0.225	0.250
10	2.146	0.270	0.744	0.319	0.253	0.191	0.201
15	2.302	0.253	0.663	0.350	0.225	0.200	0.110
20	2.330	0.250	0.629	0.378	0.266	0.196	0.165
25	2.327	0.250	0.641	0.355	0.281	0.223	0.170
30	2.302	0.253	0.646	0.369	0.271	0.234	0.164
35	2.307	0.254	0.650	0.362	0.275	0.221	0.164

Table A.16: ChannelNet **2-channel** model @ SINR 10 dB - configuration 32x2

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.265	0.393	1	0.743	0.487	0.170	0.118
0	2.106	0.313	0.955	0.176	0.066	0	0.087
5	3.353	0.157	0.428	0.005	0	0	0.085
10	4.283	0.050	0.113	0	0	0	0.101
15	4.657	0.009	0.018	0	0	0	0.081
20	4.718	0.002	0.004	0	0	0	0.082
25	4.726	0.001	0.002	0	0	0	0.081
30	4.731	0.001	0.001	0	0	0	0.081
35	4.731	0.001	0.001	0	0	0	0.081

Table A.17: ChannelNet **multi-channel** model @ SINR 10 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.215	0.396	1	0.770	0.528	0.185	0.142
0	2.083	0.321	0.956	0.219	0.049	0	0.118
5	2.778	0.221	0.661	0.032	0.040	0	0.124
10	3.392	0.151	0.387	0.043	0.020	0	0.098
15	3.628	0.124	0.305	0.040	0.017	0	0.094
20	3.672	0.119	0.290	0.041	0.022	0	0.099
25	3.664	0.120	0.292	0.046	0.025	0	0.098
30	3.588	0.130	0.320	0.046	0.035	0	0.099
35	3.557	0.134	0.334	0.036	0.030	0	0.099

Table A.18: ChannelNet **2-channel** model @ SINR 20 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.250	0.393	1	0.742	0.505	0.186	0.112
0	2.129	0.311	0.950	0.175	0.032	0	0.122
5	3.305	0.162	0.443	0.013	0	0	0.108
10	4.316	0.047	0.104	0	0	0	0.084
15	4.660	0.009	0.017	0	0	0	0.081
20	4.718	0.002	0.004	0	0	0	0.080
25	4.731	0.001	0.001	0	0	0	0.080
30	4.733	0.000	0.001	0	0	0	0.080
35	4.731	0.001	0.001	0	0	0	0.080

Table A.19: ChannelNet **multi-channel** model @ SINR 20 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.092	0.403	1	0.817	0.545	0.315	0.279
0	1.978	0.335	0.976	0.303	0.061	0.020	0.120
5	2.607	0.245	0.739	0.051	0.040	0	0.114
10	3.262	0.168	0.448	0.052	0.022	0	0.114
15	3.514	0.139	0.350	0.040	0.019	0	0.113
20	3.565	0.132	0.329	0.047	0.029	0	0.090
25	3.524	0.138	0.345	0.056	0.031	0	0.091
30	3.430	0.149	0.383	0.060	0.033	0	0.092
35	3.371	0.156	0.406	0.062	0.033	0	0.092

Table A.20: ChannelNet **2-channel** model @ SINR 50 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.235	0.394	1	0.760	0.468	0.216	0.140
0	2.080	0.317	0.961	0.201	0.036	0	0.105
5	3.269	0.167	0.457	0.014	0	0	0.113
10	4.252	0.055	0.120	0	0	0	0.102
15	4.644	0.010	0.021	0	0	0	0.081
20	4.733	0.000	0.001	0	0	0	0.081
25	4.733	0.000	0.001	0	0	0	0.079
30	4.736	0	0	0	0	0	0.079
35	4.733	0	0.001	0	0	0	0.079

Table A.21: ChannelNet **multi-channel** model @ SINR 50 dB - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.960	0.406	1	0.869	0.593	0.369	0.276
0	1.854	0.343	0.979	0.381	0.108	0.105	0.144
5	2.406	0.266	0.803	0.107	0.055	0.070	0.136
10	2.959	0.203	0.561	0.082	0.100	0	0.098
15	3.109	0.183	0.481	0.104	0.056	0	0.111
20	3.142	0.178	0.454	0.163	0.055	0	0.118
25	3.114	0.181	0.470	0.159	0.042	0	0.127
30	3.071	0.185	0.487	0.175	0.043	0	0.132
35	3.025	0.192	0.509	0.182	0.047	0.005	0.135

Table A.22: 2D-MMSE standard algorithm - 32x2 configuration

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.151	0.399	1	0.820	0.544	0.176	0.118
0	2.096	0.313	0.959	0.185	0.032	0	0.121
5	3.295	0.162	0.446	0.014	0	0	0.108
10	4.308	0.048	0.105	0	0	0	0.084
15	4.672	0.007	0.014	0	0	0	0.081
20	4.728	0.001	0.002	0	0	0	0.081
25	4.733	0	0.001	0	0	0	0.080
30	4.736	0	0	0	0	0	0.080
35	4.733	0	0.001	0	0	0	0.080

Appendix B

NR-CsiNet Simulation Results

B.1 CSI Feedback compression in 8x2 MIMO scenario

Table B.1: 8×2 NR-CsiNet **2-channel** model - encoded dim 80, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.281	0.500	1	1	0.957	0.766
0	6.364	0.500	1	0.733	0.297	0.060
5	9.571	0.490	0.972	0.080	0	0
10	14.905	0.238	0.376	0	0	0
15	19.665	0.013	0.016	0	0	0
20	19.958	0.000	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.2: 8×2 NR-CsiNet **2-channel** model - encoded dim 40, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.311	0.500	1	1	0.957	0.759
0	6.394	0.500	1	0.728	0.298	0.054
5	9.581	0.490	0.972	0.078	0	0
10	14.885	0.240	0.377	0	0	0
15	19.665	0.013	0.016	0	0	0
20	19.958	0	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.3: 8×2 NR-CsiNet **2-channel** model - encoded dim 20, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.321	0.500	1	1	0.957	0.757
0	6.333	0.500	1	0.741	0.298	0.071
5	9.581	0.490	0.973	0.079	0	0
10	14.926	0.237	0.373	0	0	0
15	19.625	0.014	0.018	0	0	0
20	19.958	0	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.4: 8×2 NR-CsiNet **2-channel** model - encoded dim 10, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.230	0.500	1	1	0.959	0.779
0	6.273	0.500	1	0.753	0.302	0.080
5	9.490	0.490	0.972	0.098	0	0
10	14.714	0.248	0.390	0	0	0
15	19.595	0.015	0.020	0	0	0
20	19.948	0.001	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.5: 8×2 NR-CsiNet **multi-channel** model - encoded dim 80, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.301	0.500	1	1	0.957	0.761
0	6.374	0.500	1	0.732	0.296	0.054
5	9.591	0.490	0.972	0.076	0	0
10	14.885	0.240	0.377	0	0	0
15	19.625	0.015	0.019	0	0	0
20	19.958	0	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.6: 8×2 NR-CsiNet **multi-channel** model - encoded dim 40, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.291	0.500	1	1	0.959	0.763
0	6.374	0.500	1	0.734	0.295	0.056
5	9.571	0.490	0.972	0.079	0	0
10	14.885	0.239	0.376	0	0	0
15	19.615	0.015	0.019	0	0	0
20	19.958	0.000	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.7: 8×2 NR-CsiNet **multi-channel** model - encoded dim 20, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.240	0.500	1	1	0.959	0.773
0	6.283	0.500	1	0.754	0.305	0.071
5	9.510	0.491	0.975	0.088	0	0
10	14.734	0.247	0.389	0	0	0
15	19.635	0.015	0.018	0	0	0
20	19.958	0.000	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

Table B.8: 8×2 NR-CsiNet **multi-channel** model - encoded dim 10, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4
-5	1.059	0.500	1	1	0.963	0.811
0	5.859	0.500	1	0.790	0.385	0.145
5	9.359	0.491	0.976	0.129	0.003	0
10	14.048	0.281	0.457	0	0	0
15	19.292	0.029	0.037	0	0	0
20	19.948	0	0.001	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

B.2 CSI Feedback estimation and compression in 8x2 MIMO scenario

Table B.9: 8×2 NR-CsiNet **2-channel** model @ SINR 10 dB - encoded dim 10, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.170	0.500	1	1	0.972	0.775	0.191
0	6.454	0.500	1	0.769	0.278	0.038	0.076
5	9.520	0.496	0.989	0.058	0	0	0.027
10	14.442	0.265	0.445	0	0	0	0.015
15	19.706	0.010	0.014	0	0	0	0.013
20	19.968	0	0	0	0	0	0.011
25	19.968	0	0	0	0	0	0.011
30	19.968	0	0	0	0	0	0.011
35	19.968	0	0	0	0	0	0.011

Table B.10: 8×2 NR-CsiNet **2-channel** model @ SINR 10 dB - encoded dim 40, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.958	0.500	1	1	0.986	0.810	0.179
0	6.172	0.500	1	0.822	0.304	0.098	0.071
5	9.480	0.498	0.993	0.065	0.025	0	0.029
10	14.431	0.265	0.425	0	0	0	0.018
15	19.615	0.015	0.018	0	0	0	0.016
20	19.928	0.002	0.002	0	0	0	0.015
25	19.968	0	0	0	0	0	0.016
30	19.968	0	0	0	0	0	0.016
35	19.968	0	0	0	0	0	0.016

Table B.11: 8×2 NR-CsiNet **2-channel** model @ SINR 10 dB - encoded dim 20, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.928	0.500	1	1	0.986	0.817	0.138
0	6.081	0.500	1	0.830	0.318	0.116	0.057
5	9.460	0.498	0.995	0.067	0.025	0	0.035
10	14.371	0.267	0.432	0	0	0	0.030
15	19.524	0,019	0.024	0	0	0	0.028
20	19.928	0.002	0.002	0	0	0	0.027
25	19.968	0	0	0	0	0	0.027
30	19.968	0	0	0	0	0	0.027
35	19.968	0	0	0	0	0	0.027

Table B.12: 8×2 NR-CsiNet **2-channel** model @ SINR 10 dB - encoded dim 10, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.918	0.500	1	1	0.986	0.819	0.379
0	6.051	0.500	1	0.837	0.329	0.116	0.148
5	9.409	0.498	0.993	0.079	0.025	0	0.073
10	14.240	0.273	0.442	0.002	0	0	0.061
15	19.474	0.021	0.027	0	0	0	0.060
20	19.928	0.002	0.002	0	0	0	0.062
25	19.968	0	0	0	0	0	0.062
30	19.968	0	0	0	0	0	0.062
35	19.968	0	0	0	0	0	0.062

Table B.13: 8×2 NR-CsiNet **2-channel** model @ SINR 20 dB - encoded dim 80, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	1.160	0.500	1	1	0.990	0.765	0.126
0	6.545	0.500	1	0.772	0.186	0.069	0.039
5	9.581	0.500	1	0.033	0.010	0	0.017
10	15.470	0.213	0.334	0	0	0	0.012
15	19.746	0.009	0.012	0	0	0	0.012
20	19.968	0	0	0	0	0	0.011
25	19.968	0	0	0	0	0	0.011
30	19.968	0	0	0	0	0	0.011
35	19.968	0	0	0	0	0	0.011

Table B.14: 8×2 NR-CsiNet **2-channel** model @ SINR 20 dB - encoded dim 40, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.988	0.500	1	1	0.986	0.804	0.097
0	6.162	0.500	1	0.820	0.307	0.100	0.035
5	9.480	0.498	0.993	0.064	0.025	0	0.016
10	14.462	0.263	0.422	0	0	0	0.012
15	19.635	0.014	0.017	0	0	0	0.011
20	19.928	0.002	0.002	0	0	0	0.011
25	19.968	0	0	0	0	0	0.011
30	19.968	0	0	0	0	0	0.011
35	19.968	0	0	0	0	0	0.011

Table B.15: 8×2 NR-CsiNet **2-channel** model @ SINR 20 dB - encoded dim 20, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.958	0.500	1	1	0.986	0.810	0.091
0	6.132	0.500	1	0.827	0.314	0.109	0.039
5	9.480	0.498	0.993	0.064	0.025	0	0.028
10	14.442	0.264	0.423	0	0	0	0.027
15	19.544	0.018	0.022	0	0	0	0.027
20	19.928	0.002	0.002	0	0	0	0.027
25	19.968	0	0	0	0	0	0.027
30	19.968	0	0	0	0	0	0.027
35	19.968	0	0	0	0	0	0.027

Table B.16: 8×2 NR-CsiNet **2-channel** model @ SINR 20 dB - encoded dim 10, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.918	0.500	1	1	0.986	0.819	1.055
0	6.061	0.500	1	0.837	0.321	0.127	0.334
5	9.429	0.498	0.993	0.076	0.025	0	0.125
10	14.250	0.272	0.445	0	0	0	0.078
15	19.423	0.023	0.029	0	0	0	0.068
20	19.928	0.002	0.002	0	0	0	0.063
25	19.968	0	0	0	0	0	0.063
30	19.968	0	0	0	0	0	0.063
35	19.968	0	0	0	0	0	0.063

Table B.17: 8×2 - Follow PMI algorithm

Follow PMI						
SINR [dB]	**Thr [Mbit/s]**	**Dec BER**	**BLER1**	**BLER2**	**BLER3**	**BLER4**
-5	0.504	0.500	1	1	0.990	0.901
0	4.962	0.500	1	0.900	0.491	0.266
5	8.905	0.500	1	0.198	0.057	0
10	12.929	0.336	0.577	0.002	0	0
15	18.939	0.046	0.057	0	0	0
20	19.918	0.002	0.003	0	0	0
25	19.968	0	0	0	0	0
30	19.968	0	0	0	0	0
35	19.968	0	0	0	0	0

B.3 CSI Feedback estimation and compression with NR-CsiVAE: 8x2 MIMO scenario

Table B.18: 8×2 NR-CsiVAE **2-channel** model @ SINR 10 dB - encoded dim 20, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.121	0.500	1	1	1	0.975	0.262
0	3.167	0.500	1	0.981	0.652	0.624	0.061
5	7.664	0.500	1	0.439	0.237	0.027	0.033
10	11.013	0.417	0.753	0.089	0	0	0.025
15	16.095	0.181	0.262	0	0	0	0.023
20	19.444	0.022	0.029	0	0	0	0.022
25	19.887	0.003	0.004	0	0	0	0.022
30	19.968	0	0	0	0	0	0.022
35	19.968	0	0	0	0	0	0.022

Table B.19: 8×2 NR-CsiVAE **2-channel** model @ SINR 10 dB - encoded dim 10, $K_{DFT} = 3$

SINR [dB]	Thr [Mbit/s]	Dec BER	BLER1	BLER2	BLER3	BLER4	σ_{NMSE}
-5	0.343	0.500	1	1	1	0.929	0.288
0	3.893	0.500	1	0.928	0.603	0.527	0.124
5	7.866	0.500	1	0.350	0.242	0.12	0.092
10	11.779	0.388	0.684	0.054	0.025	0	0.060
15	17.043	0.136	0.177	0.007	0	0	0.053
20	19.302	0.027	0.036	0	0	0	0.053
25	19.786	0.006	0.009	0	0	0	0.052
30	19.847	0.004	0.006	0	0	0	0.052
35	19.887	0.003	0.004	0	0	0	0.051

Abbreviations and Acronyms

*s*TTI	*shortened* Transmission Time Interval
1D	1-dimensional
2D	2-dimensional
3D	3-dimensional
AAS	Active Antenna System
AI	Artificial Intelligence
ALMMSE	Approximated Linear Minimum Mean Square Error
AMC	Automatic Modulation Classification
AS	Angle Spread
BER	Bit Error Rate
BGD	Batch Gradient Descent
BLER	BLock Error Rate
BP	Bandwidth Part
BS	Base Station
CA	Carrier Aggregation
CC	Component Carrier
CDL	Cluster Delay Line
CDM	Code Division Multiplexing
CDM group	Code Division Multiplexing group
CDMA	Code Division Multiple Access
CFR	Channel Frequency Response
CIR	Channel Impulse Response
CNN	Convolutional Neural Network
CoMP	Coordinated Multi-Point
CP	Cyclic Prefix

CP-OFDM	Cyclic Prefix Orthogonal Frequency-Division Multiplexing
CQI	Channel Quality Indicator
CRBs	Common Resource Blocks
CRI	CSI-RS Resource Indicator
CS	Compressive Sensing
CSI	Channel State Information
CSI-RS	Channel State Information-Reference Signal
CsiNet	CSI Network
DCI	Downlink Channel Information
DCT	Discrete Cosine Transform
DetNet	Detection Network
DFT	Discrete Fourier Transform
DL	Downlink
DL	Deep Learning
DM-RS	DeModulation-Reference Signal
DnCNN	Denoising Convolutional Neural Network
DNN	Deep Neural Network
DOA	Direction of Arrival
e-NodeB	enhanced-NodeB
EMBB	Enhanced Mobile BroadBand
FD-MIMO	Full Dimensional MIMO
FDD	Frequency Division Duplexing
FDM	Frquency Division Multiplexing
FFT	Fast Fourier Transform
gNodeB	NR base station
H-ARQ	Hybrid-Automatic Repeat reQuest
HR	High Resolution
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
IR	Image Restoration
ISI	InterSymbol Interference

KL	Kullback-Laibler
LI	Layer Indicator
LOS	Line Of Sight
LR	Low Resolution
LS	Least Squares
LSTM	Long Short-Term Memory
LTE	Long Term Evolution
LTE-A	LTE-Advanced
LVM	Latent Variable Model
MAC	Medium Access Control
MAC CE	MAC Control Element
Massive MIMO	Massive Multiple-Input Multiple-Output
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MMSE	Minimum Mean Square Error
MMTC	Massive Machine-Type Communication
mmWave	millimeter Wave
MSE	Mean Square Error
MU-MIMO	Multi-User Multiple-Input Multiple-Output
NB-IoT	Narrowband Internet of Things
NFFT	Number of FFT samples
NG	Next Generation
NLOS	Non-Line Of Sight
NMSE	Normalized MSE
NN	Neural Network
NOFDM	Number of OFDM symbols per TTI
NOMA	Non-Orthogonal Multiple Access
NPRB	Number of PRB
NR	New Radio
NSUB	Number of SUBcarriers used
OFDM	Orthogonal Frequency-Division Multiplexing
PCA	Principal Component Analysis

PDSCH	Physical Downlink Shared CHannel
PMI	Precoding Matrix Indicator
PRB	Physical Resource Block
PUCCH	Physical Uplink Control CHannel
QoS	Quality of Service
RB	Resource Block
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
RE	Resource Element
ReLU	Rectified Linear Unit
RI	Rank Indicator
RNN	Recurrent Neural Network
RS	Reference Signals
SC	Sparse-Coding
SFI	Slot Format Indicator
SGD	Stochastic Gradient Descent
SINR	Signal to Noise plus Interference Ratio
SISO	Single-Input Single-Output
SNR	Signal-to-Noise Ratio
SR	Super-Resolution
SRCNN	Super-Resolution Convolutional Neural Network
SS	Subcarrier Spacing
SSBRI	Synchronization Signal/PBCH Block Resource Indicator
SU-MIMO	Single-User Multiple-Input Multiple-Output
SVD	Singular Value Decomposition
TBS	Transport Block Size
TDD	Time Division Duplexing
TDL	Tapped Delay Line
TDM	Time Division Multiplexing
TTI	Transmission Time Interval
TXRU	Transceiver Unit
UE	User Equipment

UL	Uplink
URLLC	Ultra-Reliable Low-Latency Communication
V2V	Vehicle-Two-Vehicle
V2X	Vehicle-to-everything
VAE	Variational AutoEncoder
VehA	Vehicular-A

Bibliography

[1] 3GPP. Nr; study on test methods. Technical Report (TR) 38.810, 3rd Generation Partnership Project (3GPP), 09 2018. Version 2.4.0.

[2] 3GPP. Nr; multiplexing and channel coding. Technical Specification (TS) 38.212, 3rd Generation Partnership Project (3GPP), 06 2019. Version 15.6.0.

[3] 3GPP. Nr; physical channels and modulation. Technical Specification (TS) 38.211, 3rd Generation Partnership Project (3GPP), 06 2019. Version 15.6.0.

[4] 3GPP. Nr; physical layer procedures for data. Technical Specification (TS) 38.214, 3rd Generation Partnership Project (3GPP), 06 2019. Version 15.6.0.

[5] E. Dahlman, S. Parkvall, and J. Skold. *5G NR: The Next Generation Wireless Access Technology*. Elsevier Science, 2018.

[6] Carl Doersch. Tutorial on variational autoencoders. 2016.

[7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[8] Guan Gui, Hongji Huang, Yiwei Song, and Hikmet Sari. Deep learning for an effective nonorthogonal multiple access scheme. *IEEE Transactions on Vehicular Technology*, 67(9):8440–8450, 2018.

[9] Hongji Huang, Song Guo, Guan Gui, Zhen Yang, Jianhua Zhang, Hikmet Sari, and Fumiyuki Adachi. Deep learning for physical-layer 5g wireless techniques: Opportunities, challenges and solutions. 2019.

[10] Hongji Huang, Yiwei Song, Jie Yang, Guan Gui, and Fumiyuki Adachi. Deep-learning-based millimeter-wave massive mimo for hybrid precoding. *IEEE Transactions on Vehicular Technology*, 68(3):3027–3032, 2019.

[11] Hongji Huang, Jie Yang, Hao Huang, Yiwei Song, and Guan Gui. Deep learning for super-resolution channel estimation and doa estimation based massive mimo system. *IEEE Transactions on Vehicular Technology*, 67(9):8549–8560, 2018.

[12] Aaron Courville Ian Goodfellow, Yoshua Bengio. *Deep learning*. Adaptive computation and machine learning. MIT Press, Cambridge (Mass.), 2016.

[13] Chunxiao Jiang, Haijun Zhang, Yong Ren, Zhu Han, Kwang-Cheng Chen, and Lajos Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Commun.*, 24(2):98–105, 2017.

[14] P. Kim. *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*. Apress, 2017.

[15] Ping-Heng Kuo, H. T. Kung, and Pang-an Ting. *Compressive Sensing Based Channel Feedback Protocols for Spatially-Correlated Massive Antenna Arrays*. Institute of Electrical and Electronics Engineers, 2012.

[16] Kevin P Murphy. *Machine learning : a probabilistic perspective / Kevin P. Murphy*. Adaptve computation and machine learning series. MIT Press, Cambridge (Mass.), 2012.

[17] Timothy J. O'Shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Trans. Cogn. Comm. & Networking*, 3(4):563–575, 2017.

[18] Sreeraj Rajendran, Wannes Meert, Domenico Giustiniano, Vincent Lenders, and Sofie Pollin. Distributed deep learning models for wireless signal classification with low-cost spectrum sensors. 2017.

[19] Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2015.

[20] Magnus Sandell and Ove Edfors. *A comparative study of pilot-based channel estimators for wireless OFDM*, volume TULEA 1996:19. 1996.

[21] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh. Deep learning-based channel estimation. *IEEE Communications Letters*, 23(4):652–655, April 2019.

[22] T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin. Deep learning for wireless physical layer: Opportunities and challenges. *China Communications*, 14(11):92–111, Nov 2017.

[23] C. Wen, W. Shih, and S. Jin. Deep learning for massive mimo csi feedback. *IEEE Wireless Communications Letters*, 7(5):748–751, Oct 2018.

[24] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. Power of deep learning for channel estimation and signal detection in OFDM systems. *IEEE Wireless Commun. Letters*, 7(1):114–117, 2018.

[25] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

Web References

[W1] http://morse.colorado.edu/~tlen5510/text/classweb229x.png.

[W2] http://www.sharetechnote.com/html/5G/5G_CSI_RS_Codebook.html.

[W3] http://www.sharetechnote.com/html/5G/5G_CSI_RS_Codebook.html.

[W4] http://www.sharetechnote.com/html/5G/5G_CSI_RS_Codebook.html.

[W5] en.wikipedia.org/wiki/Feature_scaling.

[W6] https://5g.co.uk/guides/what-is-5g-new-radio/.

[W7] https://adventuresinmachinelearning.com/stochastic-gradient-descent/.

[W8] https://chsasank.github.io/deep-learning-crash-course-2.html.

[W9] https://deepakbaby.github.io/post/vae-insights/.

[W10] http://sharetechnote.com/html/5G/image/5G_MassiveMIMO_FDMIMO_01.png.

[W11] https://iaml.it/blog/variational-autoencoders-1.

[W12] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-. neural-network-cnn-deep-learning-99760835f148.

[W13] https://m.eet.com/media/1246332/5G_usage_cases_ITU_2016.png.

[W14] https://tiao.io/post/tutorial-on-variational-autoencoders-with-a-. concise-keras-implementation/.

[W15] https://towardsdatascience.com/applied-deep-learning-part-1-. artificial-neural-networks-d7834f67a4f6.

[W16] https://towardsdatascience.com/how-to-teach-a-computer-to-. see-with-convolutional-neural-networks-96c120827cd1.

[W17] https://towardsdatascience.com/part-4-better-faster-stronger-.dd6ded07b74f.

[W18] https://wireless.engineering.nyu.edu/wp-content/uploads/2018/04.

[W19] https://www.hindawi.com/journals/mpe/2016/7616393/.

[W20] https://www.kdnuggets.com/2018/03/5-things-reinforcement-.learning.html.

[W21] https://www.nature.com/articles/s41598-017-10324-y.

[W22] https://www.rfwireless-world.com/images/5G-NR-DL-UL-Frame.jpg.

[W23] https://www.rfwireless-world.com/images/5G-NR-Frame.jpg.

[W24] https://www.rfwireless-world.com/images/5G-NR-slots-per-subframe.jpg.

[W25] https://www.sciencedirect.com/topics/engineering/multiple-antenna.

[W26] http://www.sthda.com/english/articles/tag/pam-clustering.