Linköping University | STIMA
Master's thesis, 30 ECTS | Statistics and Machine Learning
2022 | LIU-IDA/STAT-EX-A--2022/001--SE

# ML-aided Cross-Band Channel prediction in MIMO systems

– ML-stödd prediktion av kanal i tvärband i Mimo-system

**Alejo Pérez Gómez**

Supervisor : Héctor Rodríguez-Déniz
Examiner : Krzystof Bartozek

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

## Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`.

## Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`.

© Alejo Pérez Gómez

## Abstract

The wireless communications technologies have experienced a exponential development during the last decades. 5G is a prominent exponent whose one of its crucial components is the Massive MIMO technology. By supporting multiple streams of signals it allows a revamped signal reconstruction in terms of mobile traffic size, data rate, latency and reliability. In this thesis work, we isolated this technology into a SIMO approach (Single-Input Multiple-Output) to explore a Machine Learning modeling to address the so-called Channel Prediction problem. Generally, the algorithms available to perform Channel Estimation in FDD and TDD deployments incur computational complexity downsides and require explicit feedback from client devices, which is typically prohibitive.

This thesis work focuses on Channel Prediction by aims of employing Machine and Deep Learning models in order to reduce the computational complexity by further relying in statistical modeling/learning. We explored the cross-Frequency Subband prediction intra-TTI (Transmission Time Interval) by means of proposing 3 three models. These intended to leverage frequency Multipath Components correlations along TTIs. The first two ones are Probabilistic Principal Components Analysis (PPCA) and its Bayesian approach, Bayesian Principal Components Analysis (BPCA). Then, we implemented a Deep Learning Variational Encoder-Decoder (VED) architecture. These three models intended to deal with the hugely high-dimensional space of the 4 datasets used by its intrinsic dimensionality reduction.

By evaluating the performance of all models, we highlight that the PPCA model outperformed the others in all the datasets, yielding a MSE error of 0.0029, 0.0112, 0.0584 and 0.0247.

# Acknowledgments

First and foremost, I would like to thank my thesis supervisor, Héctor Rodrguez-Déniz of Linköpings University's Division of Statistics and Machine Learning (STIMA) of the Department of Computer and Information Science (IDA). When I encountered difficulties or was struggling with my research or writing, the door to Prof. Rodrguez-Déniz's office was always open. He consistently let me do my own work on this study, but steered me in the right direction when he thought I needed it.

I would also like to thank the all the experts who were involved in the course of this research project from The R&D Center of Huawei Technologies Sweden AB. Specially Jinliang Huang, Karl Gafvert, Olivier Verdier and Anestis Tsakmalis. Without their passionate participation, inputs and expertise, the work carried out could not have been successfully conducted.

I would also like to acknowledge Krzysztof Bartoszek of the STIMA division at Linköpings Universitet as the examiner of the thesis, as well as to Siddharth Swaminathan for being my opponent. I am very much grateful for their insightful comments on this work.

Finally, I must express my very profound gratitude to my parents, girlfriend and friends for always providing unwavering support and encouragement throughout my years of study and the process of researching and writing my thesis. This achievement would not have been attained without your assistance. Thank you.

Alejo Pérez Gómez

# Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation

The exponential development in the field of wireless communications during the last decades has generated a well-established trend in the research of 5G, and WiFi technologies. To meet the 5G requirement demands in terms of performance, data-rate, spectral efficiency and network capacity, Multiple-Input Multiple-Output (MIMO) is promoted as a promising field of research to be explored. Such strategies aim at increasing the throughput and capacity of wireless networks addressing the issue of multi-antenna approaches, thus taking advantage of more data streams and signal quality [1].

A common wireless transmission context is encompassed in a imaginary portion of the land named Cellular Network (Cell). This, in turn, includes a Base Station (BS) with a geometrically-arranged number of antennas (in the order 64 to 128 for massive-MIMO). This BS communicates wirelessly with different user equipment (UE) typically in movement. The electromagnetic (EM) signal sent is affected in multiple ways when traversing the wireless environment in $BS \rightarrow UE$ direction (called Downlink), and $UE \rightarrow BS$ (called Uplink) [2].

Characterising the effect of this wireless environment in the radio signal, also known as Channel, is a core problem addressed in the realm of telecommunications, the so-called Channel Estimation. In addition, when applied in multiple-antenna scenarios as such, this procedure generally leads to high overheads and computational complexity, therefore is deemed a prohibitive task in its related scientific literature. To circumvent this drawback, numerous strategies have been studied using a wide variety of approaches (see Literature Review Section 2) among which we highlight the Channel Prediction through Statistical Modeling or Machine Learning (ML) techniques. The Channel Prediction approach intends to save computational resources and overcome the limiting time constraints of numerical optimisation techniques. The R&D Center of Huawei Technologies Sweden AB is regarded as prominent in such a research domain, being selected by ICT China 2021 as the "Best Solution Case" at the PT Expo China (PTEXPO) 2021 for its outstanding network performance and extensive commercialisation of indoor distributed Massive MIMO [3]. This thesis is done in collaboration with Huawei Technologies Sweden AB due to their long-standing interest in applying ML techniques in pursue of a revamped Channel Prediction.

## 1.2 Background

The aforementioned wireless signal undergoes fading, that is to say, random variations in the attenuation of the signal transmitted. This is generated by the Multipath Propagation phenomenon (among others like power decay), that makes the signal arrive by different separate subpaths caused by the reflection and refraction against geographical elements (scatterers) in its way. The Multipath Propagation explains shifts in phase (time delays) and amplitude (interferences) of the radio signal, specified as $\tau$ and $\alpha$ respectively in Figure 1.1, known as Multipath Components (MPC). This issue is of utmost importance in our thesis, because when the signal is received in BS, the different subpaths can be leveraged to reconstruct the original signal with more quality. Note that some received signal paths can actually be the resulting combination of different subpaths that arrive simultaneously. This capacity of the BS of being able to discern between more similar subpaths depends on its Spatial Resolution.

In our context, this scenario will be tackled with a Single-Input Multiple-Output (SIMO) approach, since our provided data accounts for one single UE (User Equipment) and a multiple antennae set ($M$ antennae) that receive the signal as BS (see Chapter 3) for complete description of the data).

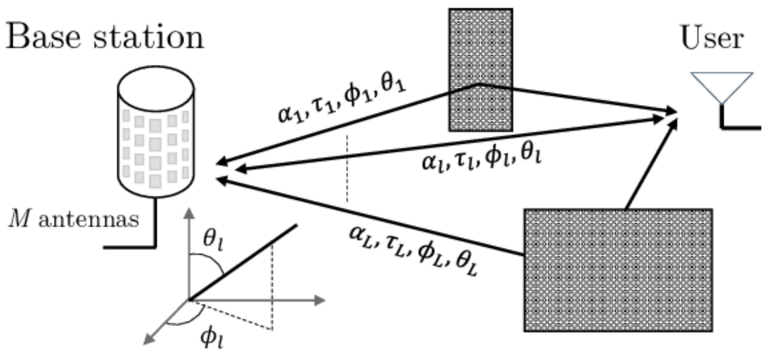

Base station
User
$M$ antennas
$\alpha_1, \tau_1, \phi_1, \theta_1$
$\alpha_l, \tau_l, \phi_l, \theta_l$
$\alpha_L, \tau_L, \phi_L, \theta_L$
$\theta_l$
$\phi_l$

Figure 1.1: MIMO Multipath Propagation Drawing
Massive MIMO multipath propagation environment described by L components. Each multipath component is characterized by its complex gain $\alpha_1$ , its delay $\tau_l$ , angle to the horizon (azimuth) $\phi_l$ and elevation angle $\theta_l$ (obtained from [4])

The so-called channel is theoretically formulated in the following vectorial form (in complex domain) of Equations 1.1 and 1.2. We can distinguish an output signal assigned to a vector $Y$, whose $y_i, ..., y_M$ elements correspond to the complex subpath signal received in each of the $M$ antennas. Then, the input signal $X$ is operated on by the channel $H$ (whose shape is analog to $Y$), with the presence of Additive White Gaussian Noise (AWGN) and further interference referred to as $n$ . This $H$ is also referred to as Channel State Information (CSI) in the literature, and describes how the signal undergoes fading, reflection on scatterers, power decay, etc. and is used to reconstruct the signal through data processing algorithms.

$$Y = HX + n \tag{1.1}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_M \end{bmatrix} x + \begin{bmatrix} n_1 \\ \vdots \\ n_M \end{bmatrix} \tag{1.2}$$

Reconstructing the input signal by performing

$$X = H^{-1}Y \tag{1.3}$$

is considered an intractable problem to target because of the shift in $H$ when exiting the coherence time[1] and the computational unfeasibility of inverting it. Hence, what is done in practise is to estimate this channel characteristics via numerical/statistical procedures. This is done by periodically estimating the channel with a signal easier to decode ($X_{pilot}$). This is to say, $X_{pilot}$ is sent out and is compared with the received $Y_{pilot}$, then we characterise the CSI . In this context, $X_{pilot}$ could appear in the form of Channel State Information Reference Signal (CSI-RS). This is CSI-RS is a type of pilot signal that BS sends to UE, allowing the UE to compute the Channel State Information (CSI) and deliver it back to the BS. This process allows the network to have their connected mobile devices to be coordinated [6], because with the CSI estimated, beamforming techniques can be applied for a better addressing of the signal as well as better decoding of $X$.

As shown in Figure 1.2, in our case, data arrive to BS in a pattern-based fashion. For each Transmission Time Interval (TTI), a CSI matrix $H_{subband_i}$ is received in one out of four Frequency subbands (SBs). We call the Frequency subband that receives $H_{subband_i}$ pilot-receiving subband. By means of undisclosed compressing algorithms, a set of Multipath components $MPC_{SB_i}$ are obtained by processing $H_{subband_i}$. That means: for each of the blue pilot-receiving subbands $SB_i$ in each TTI , we are provided with a set of $MPC_{SB_i}$ of size $L$ (see Figure 1.3).

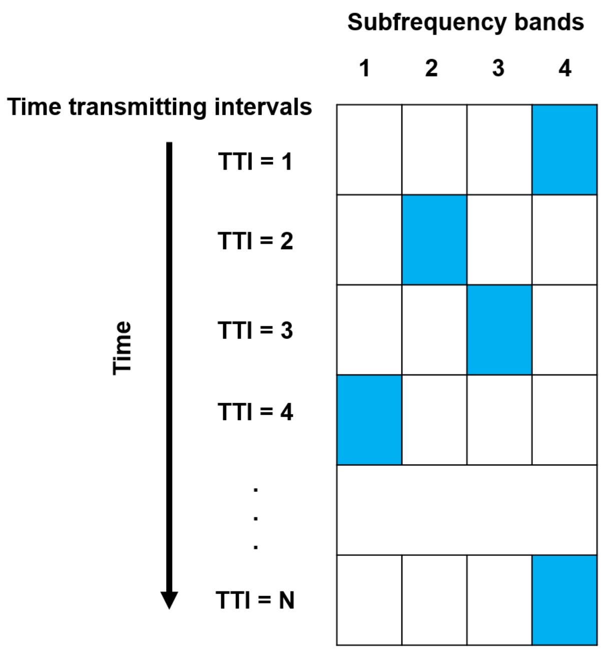


Figure 1.2: Shape of the data in a real transmission scenario (own elaboration)
The blue tiles represent the pilot-receiving Frequency Subbands whereas the white tiles correspond to the pilot-missing subbands (for each TTI). Each row of data (TTI) contains $L \times 4$ Multipath Components, belonging to the four different SBs.

## 1.3 Aim

The main objective of this thesis work is to train a model that serves us to predict the $MPC_{SB_i}$ outside the pilot-receiving subband for each time sample TTI. This is to be performed by leveraging the correlations in the frequency domain (across frequency subband) and/or time

[1]Period of time that can be calculated where the channel is considered to vary smoothly due to the absence of quick and drastic changes in the physical environment [5].

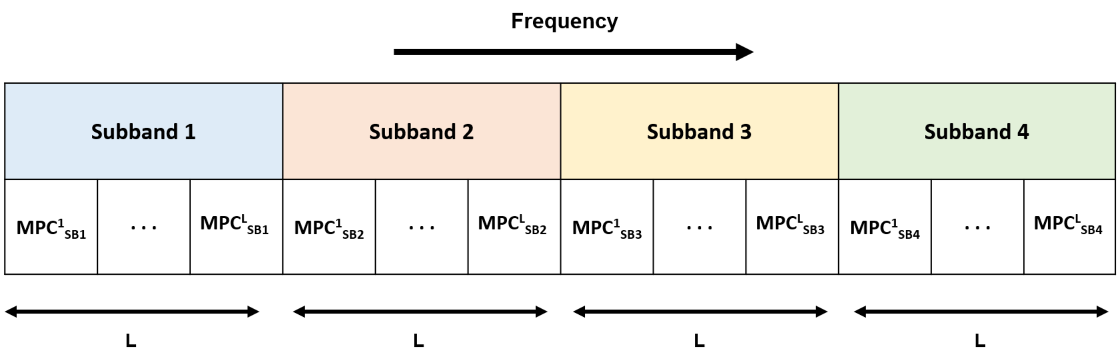


Figure 1.3: Shape of one sample TTI of the data (own elaboration)

domain samples (across TTI). Predicting the three missing $MPC_{SB_i}$ for each TTI leads to the reconstruction of the signal, the analog missing $H_{subband_i}$. We want to rely on Machine Learning and Statistical Modeling of the data in order not to incur complex and slow channel estimation algorithms. Building upon this core objective, we derive other sub-objectives:

- Characterisation of a predictive distribution $p(x_{missing} \mid x_{observed}, \theta)$. $x_{observed}$ are the observed $MPC_{SB_i}$ we train the model with (in pilot-receiving subbands), whereas $x_{missing}$ are the $MPC_{SB_i}$ we want to predict (in pilot-missing subbands). Finally, $\theta$ are the trained parameters of the ML-model. In overall, performing a cross-band prediction intra-TTI. Some sub-objectives arise from this main one:
  - Visually inspect the predictions to evaluate if the distribution of the reconstructed data has been captured properly.
  - Reconstruct the TTI samples leveraging also the correlation across time (inter-TTI).
  - Evaluate the prediction with several error metrics:
    * Assess the quality of the prediction with statistical-based error metrics
      · MSE (Mean Squared Error)
      · RMSE (Root Mean Squared Error)
      · MAE (Mean Absolute Error)
    * As the predicted CSI is used in this uplink stage for the further improvement of downlink transmission, usage of downlink reconstruction transmission metrics to assess the quality of the prediction (with a reconstructed $H$ matrix):
      · Beamforming gain
      · Network capacity
      · SNR (Signal-to-Noise ratio) gain

## 1.4 Delimitations

One foreseen delimitations is the amount of data at our disposal. We have to deal with such a high dimensionality feature-wise, having in some datasets 100 times more features than datapoints. For this reason we consider the main drawback is the data quantity, being a more ideal scenario having 10 times more datapoints than features.

Moreover, the UEs simulated in our datasets are also punctual and singular. That means that their intrinsic simulated data distributions are coming from very stochastic simulated processes that deal with their unique parametrizations. Therefore, we address our research to these targeted users but a greater scale study could comply with more generality with users coming from a wider variety of simulated distributions.

From a benchmarking point of view, another limitation of the work is that it is not possible to fully compare the results of executions carried out with the existing related literature. That is because the data used in this thesis comes from signal compression techniques and instead of CSI data we deal with Multipath Components. The presence of this approach in the literature is not actually identifiable. Let alone the fact that the studies available in the literature use engineering metrics that require back-to-back signal processing and complex algorithms for the final signal reconstruction that go far beyond of the scope of a statistical evaluation of the models. This is because we center our work in the reconstruction of Multipath Components and not in the posterior stages of signal reconstruction, which in literature serves as the eventual gold standard metric for replicability and reproducibility.

# 2 Literature Review

## 2.1 Related Work

During this chapter we firstly present a thorough literature review to describe how the related research works have tackled this issue of channel prediction. Various authors have been working in this matter from different approaches and techniques.

### 2.1.1 Signal processing approach

There exist a non-ML classical approach that tries to predict future channel CSI relying on signal processing techniques based on numerical/statistical optimization. In this regard, Swarun [7] introduced R2-F2, a system that enables LTE base stations to infer the downlink channels to a UE by observing the uplink channels from it through optimisation techniques. It is regarded as the State-of-the-Art (SOTA) when it comes to channel prediction by derivation from uplink to downlink without feedback intended in FDD MIMO systems. Wong & Evans [8], drew on non-iterative eigen-analysis technique Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT).

Peng et al. [9] used a first-order Taylor expansion-based channel prediction model to characterise the channel. They also proposed a prediction scheme that features estimation and prediction sections and thus deriving interval of effective predictions (IEP). They evaluate the model using MSE varying SNR.

In general, techniques such as Multiple Signal Classification (MUSIC) and Estimation of Signal Parameters by Rotational Invariance Techniques (ESPRIT) lack of generality and applicability as they are sensitive to the change of the data and hardware characteristics. Their estimation process is generally tedious and computationally complex due to the manipulation of high dimensional matrices. Finally, their obtained estimates last quickly due to variant nature of propagation environments with moving UEs [10].

### 2.1.2 ML approach

### 2.1.3 Cross-Temporal prediction

The attempt of channel prediction across time is another among the most wide-spread approach in the literature review collected. We can distinguish the scientific papers by the nature of the models used. Regarding time series model approach, we should point out the Auto-regressive (AR) and Kalman Filtering-related scientific works, sometimes in combination with DL or ML. The mentioned techniques in this subsection are regarded as the SOTA in this research track. Jiang & Schotten [11] aim at predicting future CSI from its past values at the same frequency/sub-carrier. They used Kalman Filter to estimate AR coefficients and also a RNN-based predictor. They evaluated the performance on computational complexity with a SNR gain of 5dB at $Pr(R) = 10^{-3}$, also with MSE at $MSE = -44.8dB$.

Liu et al. [10] made usage of Fully Connected Recurrent Neural Networks (FCRNNs) for narrow-band channel prediction. They used three different algorithms: the Real Time Recurrent Learning (RTRL), the Global Extended Kalman filter (GEKF) and the decoupled extended Kalman filter (DEKF) in order to train the Recurrent Neural Network (RNN) based channel predictor. GEKF and DEKF proved to converge faster than RTRL.

Kim et al. [12] developed and compared a vector Kalman Filter (VKF)-based channel predictor and a Machine Learning (ML)-based channel predictor using the realistic channels from the spatial channel model (SCM). The VKF-based channel predictor exploited the Auto-Regressive (AR) parameters estimated from SCM channels based on the Yule-Walker equations. Then, a MLP-based predictor was trained with the already processed data. Performance of both approaches was assessed through NMSE over different SNR. VKF's complexity was generally lesser, however when the MLP model was trained its complexity was inferior.

There also exist a well established approach where the reliance of time series forecasting on Neural Networks is stronger. In this sense, Lemayian & Hamamreh [13] proposed a Recurrent Neural Network (RNN) with Long-Short Term Memory cells (LSTM). They aimed at demonstrating low complexity channel predictions. With a similar approach but with more complex architectures, Wang et al. [14] proposed a Convolutional Long Short Term Memory Network (ConvLSTM-net)-based method. Featuring a 5x5 patch, this architecture achieved around $NMSE = -37$ dB in both time and frequency domain. In this line, Yuan et al. [15] combined pattern extraction implemented via a convolutional neural network (CNN), and a CSI predictor realized by an auto-regressive (AR) or an ARNN (Autor-regressive Neural network) with exogenous inputs recurrent neural network. In the first stage the pattern is identified and the latter step yields the prediction. They evaluated models performances through NMSE over $P$, order of AR parameters.

Some other works addressed this problem deviating from the AR modeling and focusing in dealing with the interactions of complex variables. For instance, Ding & Hirose [16] presented a method for predicting time-varying channels by combining a multilayer complex-valued neural network (CVNN) with the chirp z-transform (CZT), resulting in a CZT-ML-CVNN-based predictor. They present the accuracy in terms of SNR vs Bit Error Rate, where CZT-ML-CVNN-based outperformed the other architectures. Similarly, Yang et al.[17] used sparse complex-valued neural network (SCNet) to leverage the uplink CSI to predict the downlink feedback motivated by the universal approximation theorem. They relied on NMSE metric over $\Delta f$ and AS (Angle Spread).

### 2.1.4 Cross-Frequency Band prediction

We should recall that our data is based in Multipath Components that comes out of data compression stage from CSI channel matrices. Identifying research works that exactly map our scenario and apply cross frequency subband channel prediction have not been entirely possible.

Although, we could collect some projects that use channel prediction across Frequency Band in Frequency Division Multiplexing (FDD) communication scheme.

Bakshi et al. [18] trained a Neural Network on a standard channel model to generate coarse estimates for the underlying variables of the channel. The NN was embedded in a system they called OptML. Their goal was to reduce the complexity of the model in an order of magnitude 10x, while maintaining similar prediction quality. A Neural Network Distance Estimator (NNDE) yielded coarse estimates for distances between MPCs, these underwent further optimization and prediction processes.

### 2.1.5 Our intuition on the Channel Prediction problem

Not related to the SIMO or MIMO domain, but still framed in the wireless communications realm, Frohle et al. elaborated on Gaussian Processes (GPs) in two different studies [19], [20]. They aimed at predicting the wireless channel between mobile agents incorporating the agents' location uncertainty. We would like to claim that the spatial-temporal correlations exploited by GSP could fare well to our problem because time and frequency correlations could be exploited. Trying to estimate such a high-dimension kernel could perhaps be computationally problematic although.

Therefore, we implemented PPCA as a baseline model for prediction due to its low-complexity covariance estimation that entails feature selection as well. Previous works like Porta [21], used PPCA for action selection in an Active Appearance -Based Robot Location framework. They concretely imputate missing values with PPCA in order to apply this modeling to depth maps for the robot to robustly be space-aware. Nyamundanda [22] used PPCA for metabolic studies where the missing value imputation was necessary by modeling the estimated covariance between the high dimensional metabolite features. The Bayesian PPCA (BPCA) was used in Oba et al. [23] for missing value imputation in gene expression profiling by the modeling of its latent feature space. It outperformed singular value decomposition and K -nearest neighbors for this same task. Jenelius & Koutsopoulos used a similar missing value inference in the Urban Network Travel Time Prediction Based task [24]. They modeled the PPCA through EM-algorithm in order to predict time delays of feature observations conditioned on the already observed information using the Gaussian conditional property.

A similar modeling was intended in our baseline method, this is to say, conditioning the unobserved (missing) MPCs corresponding to the pilot-missing frequency subbands on the pilot-receiving ones. Therefore, our PPCA model is to be depicted in Section 4.1.

Aiming to use a more powerful non-linear modeling, we wanted to follow the approaches of Linfu et al. [25] and Hussein et al. [26]. These works incorporated Variational Autoencoders (VAEs) to address the downlink CSI obtention problem in a frequency division duplexing (FDD)-MIMO setting. The uplink CSI can be obtained by means of Channel Estimation while the downlink CSI should be sent to the BS from the UE, which incurs a huge overhead. These works intend to solve the problem addressed to learn an optimal/accurate compression of this downlink CSI.

Likewise, we implemented a Variational Encoder-Decoder (VED) architecture pursuing to learn an optimal reconstruction mapping as described previously for the PPCA baseline model. In short, we aimed to estimate a proper latent feature space in order to reconstruct the missing MPCs we receive at the BS was meant (note these MPCs come from prior compressing stages). Thus, for this reconstruction, we need to leverage the observed information of the pilot-receiving frequency band to later extrapolate to the missing ones. This model is described in Section 4.4. We based our model implementation in Liu et al. [27], where they used a similar modification of a VAE to predict the downlink CSI from the uplink CSI in a FDD system. Note what we used data of another nature because we dealt with MPCs resulting from further signal processing of the uplink's CSI. They achieved a SNR improvement over 10 dB in MIMO

transmissions compared to the current state-of-the-art in this regard, the R2-F2 technique [7]
.

# 3 Data

This chapter outlines the data handled in our thesis work. In the Section 3.1 the data obtained is described and in Section 3.2 the pre-processing carried out is explained. In addition, we explain how further hold-out splitting and data masking is performed for the following experiments, see Section 3.3.

## 3.1 Data obtention

The data used in this thesis consist of three different datasets. All of them where produced by simulations carried out by the Huawei Technologies Sweden AB 's Wireless Algorithm Lab. All three datasets are three-dimensional tensors [L x SB x TTI]. The parameter TTI corresponds to Transmission Time Intervals. They are called intervals because these are the duration of the transmission which is related to the encapsulation of data in blocks for an efficient communication. In practice we can treat this dimension as simple time stamps where we receive the data in a time series fashion. $SB$ stands for Frequency Subbands, these are always fixed to 4. $L$, in turn, stands for the number of Multipath Components per band, which are complex signal values.

The effective datasets used in the model executions are the ones specified in Table 3.1. We used a dataset called UE0 (90 TTIs), and two other datasets named UE1 and UE2, with 590 TTIs each. All the last 3 mentioned correspond to 3 fictional User Equipments. However, the forth dataset named JointDS is the result of the merge of UE1 and UE2. The details the dataset arrangements are specified in Section 3.2 and 3.3.

Table 3.1: Description of datasets

| **Dataset** | **Transmisison Time intervals** | **Multipath Components / SB** | **Subbands** |
|---|---|---|---|
| | TTI | MPCs / SB | SB |
| UE0 | 90 | 309 | 4 |
| UE1 | 590 | 640 | 4 |
| UE2 | 590 | 640 | 4 |
| JointDS | 1180 | 640 | 4 |

## 3.2 Data pre-processing

In order to pre-process the data for it to be fitted to a learnable ML model, we created some scripts in `R` Language to unfold these 3-dimensional tensors into 2-dimensional matrices. We concatenated each [TTI x L] slice of the tensors we were provided (horizontally) so as we got resulting matrices of 2 dimensions [TTI x L*4]. Then, we separated the real part from the imaginary one, yielding matrices of [TTI x L*4*2]. The last mentioned unfolding process (separating $Re$ from $Im$) is intended for the imaginary part of every data-point to be treated as a real number by the model. This process leads us to end up with such a large feature dimensionality. This issue is intended to be tackled by the intrinsic dimensionality reduction of our models deployed. In order to train the models, we split the matrix row-wise in a 60% / 40% fashion (Training Set/ Validation Set).

Table 3.2: Description of unfolded of datasets

| **Dataset** | **Transmisison Time intervals** | **Features** |
|---|---|---|
| | TTI | 4 SB Concatenated Horizontally and Unfolded |
| UE0 | 90 | 2472 |
| UE1 | 590 | 5120 |
| UE2 | 590 | 5120 |
| JointDS | 1180 | 5120 |

## 3.3 Masking and hold-out split

As described in Section 1.2, in a real transmission situation, every TTI of data is sensed in such a way we can only observe one out of four Frequency Subbands. This masking occurs in a well-defined fixed pattern, which in this work we decided to set as "$1 \rightarrow 3 \rightarrow 2 \rightarrow 4$" (these are the observed subbands at each TTI) . For this reason, by scripting (in R), we applied masking in 2 different ways depending on what model we made use of. For both of the two models, the dataset hold-out splitting was applied respecting the time sequence, leaving the last 40% of TTIs for the validation set.

Table 3.3: Description of hold-out split

| **Dataset** | **Training** | **Validation** |
|---|---|---|
| | TTIs | TTIs |
| UE0 | 54 | 36 |
| UE1 | 354 | 236 |
| UE2 | 354 | 236 |
| JointDS | 708 | 472 |

For the training and validation set of the JointDS, we just merged the respective training and validation sets of UE1 and UE2, hence respecting the time-line stream in which the samples where produced in a real-life situation (not shuffling TTIs between training and validation splits for none of UE1 and UE2).

### 3.3.1 Masking for PPCA model

None of the training datasets are masked when it comes to training the PPCA model. This decision in meant to assure that we are extracting proper $W$ loadings that encapsulate the greatest variance in the latent space well enough. After that, we masked the validation sets in the pattern described in the previous point in order to evaluate the predictive performance leveraged from the PPCA's obtained $W$factor loadings.

### 3.3.2 Hold-out for Variational Encoder-Decoder model

For our implemented Variational Encoder-Decoder model, we masked both training and validation sets. This whole dataset masking is intended to force a supervised learning, creating $(x_i, y_i)$ tuples that were fed to the DL architecture. For each TTI, $x_i$ corresponds to the Multipath Components in the observed Frequency Subband, (25% of the TTI), treated as predictors. The target features, collected in $y_i$, correspond to the remaining 75% of the TTI, and they are the unobserved MPCs corresponding to the three unobserved Frequency SBs. These $(x_i, y_i)$ are equivalent to the notation $(X_{obs}, X_{mis})$ that will be used in the Section 4.1 on PPCA theory.

The training set for every dataset is normalised to be fed into the VED model. That means subtracting the mean and dividing by the standard deviation (STD) feature-wise. We applied normalization in the validation set according to the mean and STD of the training set in order to prevent data leakage.

The Table 3.4 summarizes the masking and hold-out properties of each subset.

Table 3.4: Description of hold-out split datasets properties

| Model | Dataset | Obs(X) / Mis(Y) | Normalization | Masking |
|---|---|---|---|---|
| PPCA | Training |  | No | No |
|  | Validation |  |  | Yes |
| VED | Training | Observed | Yes | Yes |
|  |  | Missing | No |  |
|  | Validation | Observed | Yes |  |
|  |  | Missing | No |  |

# 4 Method

In this chapter, we proceed to depict the theoretical background of the models used in the thesis project. In the Sections 4.1 and 4.4, PPCA and VED are described. Note that the first subsections of the Section 4.4 constitute a preliminary Deep Learning framework explanation to make the background considerations of VAEs/VED more feasible for the reader. In Chapter 5, we describe how the experiments of PPCAs and VED were carried out, as well as the configured experimental set-up.

## 4.1 Probabilistic Principal Component Analysis (PPCA)

This point is to outline the two baseline models used in the thesis (PPCA and BPCA). In order to describe these models, we should first outline the Factor Analysis method for an intuitive build-up on this matter.

### 4.1.1 Factor Analysis (FA)

Let us assume a Gaussian vector $x_i$ of dimension $S \times 1$, assume $x_i \sim \mathcal{N}(\mu, \Sigma)$. We can consider $S$ to be the size of the total $MPCs$ accounting for all bands (total features per row). Now let

$$x_i = \mu + W z_i^T + \varepsilon_i, \tag{4.1}$$

where $\mu$ is the multivariate mean of $x_i$ $[S \times 1]$, $\varepsilon$ is the error term of dimension $[S \times 1]$ and $\varepsilon \sim \mathcal{N}(0, \Psi)$, and $\Psi$ is the variance of the noise. $z_i$ and $W$ have dimensions $[Q \times 1]$ and $[S \times Q]$ respectively, where $Q$ is a parameter of design choice (number of factors). $W$ is also known as the loading vector of all the features $S$ (MPCs) for all the factors. Meanwhile, $z_i$ is the score value of each of the features of $x_i$ for each factor.

Conditioning the observations on the rest of the parameters, $x_i$ can be modeled as

$$x_i \mid z_i \sim \mathcal{N}(\mu + W z_i, \Psi) \tag{4.2}$$

where we assume

$$z_i \sim \mathcal{N}(0, I). \tag{4.3}$$

Then, integrating the latent variable $z$ out we yield

$$x_i \sim \mathcal{N}(\mu, WW^T + \Psi) \tag{4.4}$$

Where $WW^T + \Psi = \Sigma$, the estimated covariance.

The general matrix for the entire dataset reads as follows

$$X = W\,Z + \mu + \varepsilon, \tag{4.5}$$

where $W$ is $[N \times Q]$, $Z$ is $[Q \times S]$, $\varepsilon$ is $[N \times S]$, $\mu$ is $[N \times S]$ and it results in the reconstructed dataset $X$ with dimensions $[N \times S]$.

## 4.2 Theoretical transition from FA to PPCA

In order to transition to PPCA, we add this constraint to the Factor Analysis model to obtain the PPCA model

$$\Psi = \sigma^2 I. \tag{4.6}$$

PPCA assumes that the data points are independent of each other given the latent variables and adds an orthogonality assumption in the factor loadings matrix $W$.

$$x_i \mid z_i \sim \mathcal{N}(\mu + W z_i^T, \quad \sigma^2 I) \tag{4.7}$$

Analogue to Factor Analysis modeling, let us integrate over $z$:

$$x_i \sim \mathcal{N}(\mu, WW^T + \sigma^2 I), \tag{4.8}$$

where $WW^T + \sigma^2 I = \Sigma$, the estimated covariance.

We assume prior $z_i \sim N(\mu_o, \Sigma_o)$ and to simplify we can promote $z_i \sim N(0, I)$. Once we have estimated $\mu$ and $W$ from the PPCA method we are interested in characterising the predictive distribution. Note first that the distribution of the latent variables given the observed data can be derived using Bayes' Theorem to give:

$$z_i \mid x_i \sim \mathcal{N}((WW^T + \sigma^2 I)^{-1} W^T (x_i - \mu), \quad \sigma^2 (WW^T + \sigma^2 I)^{-1}) \tag{4.9}$$

### 4.2.1 Dealing with missing data through inference on latent variables

Once the Probabilistic Principal Components (PPCs) are fitted, we followed the approach in Jenelius & Koutsopoulos [24] to attain a predictive distribution/function. For each sample $x_i$ ($[S \times 1]$) we divide the Multipath Components into $x_{observed}$ ($x_{obs}$) and $x_{missing}$ ($x_{mis}$). We distinguish between 2 vectors of indices that discern between the observed features and the missing ones (namely $Idx_{obs}$, $Idx_{mis}$ with dimensions $[S_{obs} \times 1]$ and $[S_{mis} \times 1]$). These correspond to the frequency subband with received pilot and the ones with no observed pilot respectively. Therefore, we define $x_{obs} = x[Idx_{obs}]$ and $x_{mis} = x[Idx_{mis}]$.

$$\mathbf{X_{obs}} = \begin{bmatrix} x_{Idx_{obs}}[1] \\ \cdot \\ \cdot \\ \cdot \\ x_{Idx_{obs}}[K_{obs}] \end{bmatrix} \quad \mathbf{X_{mis}} = \begin{bmatrix} x_{Idx_{mis}}[1] \\ \cdot \\ \cdot \\ \cdot \\ x_{Idx_{mis}}[K_{mis}] \end{bmatrix} \tag{4.10}$$

Then, the conditional property of loading matrix $W$ let us split it into the $[S_{obs} \times Q]$ matrix $W_{obs}$ and the $[S_{mis} \times Q]$ matrix $W_{mis}$. Similarly, we apply the same to $\mu$, splitting it into $\mu_{obs}$ and $\mu_{mis}$ (where $Q$ is the number of PPCs extracted).

We use the Multivariate Normal Distribution to obtain our estimates $x_{mis} \mid x_{obs} \sim \mathcal{N}(\hat{x}_{mis|obs}, \Sigma_{mis|obs})$, such that

$$\hat{x}_{mis|obs} = \mu_{mis} + W_{mis}(W_{obs}^T W_{obs} + \sigma^2 I)^{-1} W_{obs}^T (x_{obs} - \mu_{obs}) \tag{4.11}$$

$$\hat{\Sigma}_{mis|obs} = \sigma^2 W_{mis}(W_{obs}^T W_{obs} + \sigma^2 I)^{-1} W_{mis}^T + \sigma^2 I \tag{4.12}$$

The mean vector $\hat{\mu}_{mis}$ serves as point estimate for the missing Multipath Components, whereas $\Sigma_{mis|obs}$ describes the variability around those predictions. $\mu_{mis}$, in turn, establishes a baseline for the prediction to be added to the product of the updated loadings $W_{mis}(W_{obs}^T W_{obs} + \sigma^2 I)^{-1} W_{obs}^T$ times the difference vector between observed MPCs and MPCs' historical means $(x_{obs} - \mu_{obs})$.

Here below, we show how the conditional Gaussian property is applied to obtain the predictive function for $\hat{x}_{mis|obs}$. We consider a generic variable $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with some missing and observed features. Note sub-indexes $_{mo}$ and $_{om}$ stand for $_{missing|observed}$ and $_{observed|missing}$ respectively.

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_{mis} \\ \boldsymbol{\mu}_{obs} \end{bmatrix} \tag{4.13}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_{mis} \\ \boldsymbol{X}_{obs} \end{bmatrix} \tag{4.14}$$

$$\Sigma = \begin{bmatrix} \Sigma_{mis} & \Sigma_{mo} \\ \Sigma_{mi} & \Sigma_{obs} \end{bmatrix} \tag{4.15}$$

It follows that we attain the following point conditional estimates for the $x_{missing}$ mean and covariance ruled by $p(x_{mis} \mid x_{obs}) \sim \mathcal{N}(x_{mis} \mid \mu_{mis|obs}, \Sigma_{mis|obs})$

$$\hat{\mu}_{mis|obs} = \mu_{mis} + \Sigma_{mo} {\Sigma_{mis}}^{-1} (x_{obs} - \mu_{obs}) \tag{4.16}$$

$$\hat{\Sigma}_{mis|obs} = \Sigma_{mis} - \Sigma_{mo} {\Sigma_{mis}}^{-1} \Sigma_{om} \tag{4.17}$$

Notice that this result is analogous to the one in Equations 4.11 and 4.12.

## 4.3 BPCA

Following the approach of Oba et al. [23], we propose Bayesian Probabilistic Principal Components (BPCA) as a extension of our baseline. BPCA promotes priors for $p(W, \mu, \sigma^2)$ over the parameters of the model. Then, effective dimensionality over the matrix $W$ is intended to be obtained by means of the introduction of a hierarchical prior $p(W \mid \alpha)$ where $\alpha = \{\alpha_1, ...\alpha_q\}$. The maximum number of $\alpha$ hyperparameters is $q = S - 1$. $\alpha_i$ controls the inverse variance of the corresponding PPC $W_i$. The larger the $\alpha_i$, the smaller the $W_i$, eventually switching it off and hence reducing latent space dimensionality.

Finally, we can derive the natural logarithm-posterior estimate of $W$, $W_{MP}$

$$ln\, p(W \mid D) = L - \frac{1}{2}\sum_{i=1}^{d-1} \alpha_i \|w_i\|^2 + const \tag{4.18}$$

According to the R library used in this thesis (`pcaMethods` [28]) BPCA combines an EM approach for PCA with a Bayesian model. In PCA, data dissimilar to the training set but close to the latent space may have the same reconstruction error. BPCA defines a likelihood function such that the likelihood for data dissimilar the training set is much lower, even if they are close to the latent space. The algorithm does not force orthogonality between loadings $W$. thus, factor loadings are not necessarily orthogonal.

## 4.4 Variational Encoder-Decoder (VED)

As for the next course of action, in order to infer to the latent representation values to approximate the probability distribution $p(X_{mis} \mid X_{obs}, \theta_{model})$ , we proposed a DL approach that consists of an Encoder-Decoder Network with a Variational Inference layer in the bottleneck. The aim of this approach is to force each TTI sample to be embedded in a continuous value of a latent space. It is inspired by Variational Autoencoders [29]. The following sections are going to serve as a summary of the Deep Learning related theory for the Variational Encoder-Decoder description to be more accessible.

### 4.4.1 Deep Learning

The Deep Learning term refers to a segment of techniques under the umbrella of Machine Learning Methods that draw on the use of Artificial Neural Networks (ANN) by using feature learning [30]. We can distinguish two main categories, supervised learning (where inputs are labeled or coupled with a explicit target) or unsupervised (unlabeled input data). ANNs, in turn, were inspired by the information flow, processing and distribution in biological systems. To give a visual example of an NN , we show the Multilayer Perceptron (MLP) in Figure 4.1.

#### 4.4.1.1 Multilayer Perceptron (MLP)

MLPs are the quintessential types of NNs, we base the ensuing explanations in Bishop's *Pattern Recognition and Machine Learning* [31]. MLPs aim at approximating a function $f^*$ in the context $y = f^*(x)$ that maps a value $x$ to $y$. Then, let the approximation function learned by the NN be $f$ in $y = f(x;\theta)$, obtained by tweaking the set of parameters $\theta$ until they are optimal for the approximation. A simple MLP might be defined as a composition of different functions, for instance $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ being chained as $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. Let us consider a MLP with two hidden layers. Let $j = 1, ..., M$ be the set of hidden units, while $k = 1, ..., M$ is the set of outputs. The input data $X$ has dimensions $i = 1, ..., D$. In a two hidden layer NN, we can characterize the overall feed-forward function as follows:

$$y(x, \theta) = \sigma\left(\sum_{j=1}^{M} \theta_{kj}^{(2)} h\left(\sum_{i=1}^{D} \theta_{ji}^{(1)} x_i + \theta_{j0}^{(1)}\right) + \theta_{k0}^{(2)}\right) \tag{4.19}$$

We can consider $\theta_{ji}^{(1)}$ to be the weight from input dimension $i$ to the hidden node $j$ in layer (1). In the intermediate output we obtain a set of $M$ linear combinations called activations: $a_j = \sum_{i=1}^{D} \theta_{ji}^{(1)} x_i + \theta_{j0}^{(1)}$. $\sigma$ and $h$ stand for activation functions. We depict graphically this function mapping in the MLP in Figure 4.1.

Activation functions are non-linear functions applied to the activations resulting from the layers. They are useful to restrict the activation values, and for the entire neural network

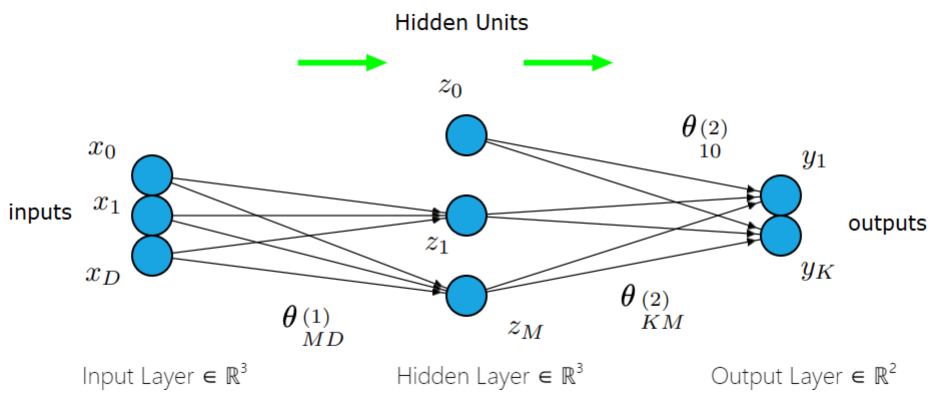


Figure 4.1: Diagram of a MLP (own elaboration)

not to collapse in a composed bigger linear function, therefore not being able to capture non-linear patterns in the data. The following equations describe different activation functions, ReLU (Equation 4.20) sets a cut-off for negative values, hyperbolic tangent (Equation 4.21) squashes a real-valued number to the range $[-1, 1]$, softplus (Equation 4.22) can be regarded as a smooth version of ReLU and LeakyReLU (Equation 4.23) is similar to ReLU but allowing small non-negative gradients for negative values of the input.

$$ReLU(\mathbf{x}) = max(0, \mathbf{x}) \tag{4.20}$$

$$tanh(\mathbf{x}) = \frac{1 - e^{(-2\mathbf{x})}}{1 + e^{(-2\mathbf{x})}} \tag{4.21}$$

$$Softplus(x) = \frac{1}{\beta}\log(1 + \exp(\beta x)) \tag{4.22}$$

$$LeakyReLU(x) = \begin{Bmatrix} & x > 0 \\ \alpha x & x <= 0 \end{Bmatrix} \tag{4.23}$$

### 4.4.1.2 Training of the Network

Neural Networks need to update their parameters so as an optimal or sub-optimal approximation is achieved by learning from a training dataset. For the sake of the explanation, we can recur to a probabilistic view of a regression problem, since the aim of the thesis is to learn a multivariate regression. Let this training dataset consist of a set of vectors $X = \{x_1...x_N\}$, being coupled with a set of target vectors $\{y_n\}$, both can take any value in real domain. We assume that $y$ has a Gaussian distribution with an x-dependent mean, which is given by the output of the neural network, see Equation 4.24 [31]. Mind $\beta$ is the inverse of the Gaussian noise (the precision). It follows that we can construct the likelihood function of the output data $y_n$ conditioned on the input data $x_n$ and the parameters (Equation 4.25). Finally, taking negative logarithm of the latter, we yield the error Equation 4.26.

$$p(y \mid x, \theta) = \mathcal{N}(y \mid f(x, \theta), \beta^{-1}) \tag{4.24}$$

$$p(y \mid X, \theta, \beta) = \prod_{n=1}^{N} p(y_n \mid x_n, \theta, \beta) \tag{4.25}$$

$$\frac{\beta}{2}\sum_{n=1}^{N}\{f(x_n,\theta)-y_n\}^2-\frac{N}{2}ln\beta+\frac{N}{2}ln(2\pi) \tag{4.26}$$

We should note that minimizing this expression in Equation 4.26 is the same as minimizing the squared term, which is analogue to minimizing the sum of squared errors.

#### 4.4.1.3 Backprogation and Adam

Let then the cost function chosen for the NN's output be $E(\theta)$, dependent on the set of parameters $\theta$. The aim is to calculate the gradient $\nabla E(\theta)$ vector that relates with the greatest increase direction. What is done in practise is decreasing in this inverse direction $-\nabla E(\theta)$ making a small step in the weight space $\theta := \theta + \nabla\theta$, where $\nabla\theta$ is the gradient vector with the corresponding magnitude to subtract to each parameter in the network. What is often carried out is gradient descent optimization, performing this update $\theta^{(\tau+1)} = \theta^{(\tau)} - \eta\nabla E(\theta^{(\tau)})$, where $\tau$ is the specific time stage and $\eta$ is the learning rate (portion of the gradient to subtract). For each of the steps performed, the gradient is calculated grouping the training data into batches, thus the resulting gradient being $E(\theta) = \sum_{n=1}^{N} E_n(\theta)$. In order to update the specific weight $\theta$, the quantity subtracted is determined by the backpropagation algorithm, described at Algorithm 1. This is to calculate the partial derivative of the weight $\theta_{ij}$ with respect to the total gradient: $\frac{\delta E}{\delta\theta_{ij}}$. Adaptive Moment Estimation (Adam) [32] is a batch-based optimisation algorithm that is computationally efficient and varies the learning rate according to the training results, see Algorithm 2.

**Algorithm 1** Backpropagation Algorithm Pseudocode [33]

```
Procedure Training
$X \leftarrow$ Training Data Set of size mxn
$y \leftarrow$ Labels for records in X
$w \leftarrow$ The weights for respective layers
$l \leftarrow$ The number of layers in the neural networks, 1...L
$D_{ij}^{(l)} \leftarrow$ The error for all l,i,j
$t_{(ij)}^{(l)} \leftarrow 0.$ For all l,i,j
for $i = 1$ to m do
    $a^l \leftarrow feedforward(x^{(i)}, w)$
    $d^l \leftarrow a(L) - y(i)$
    $t_{ij}^{(l)} \leftarrow t_{ij}^{(l)} + a_j^{(l)} \cdot t_i^{l+1}$
end for
if $j \neq 0$ then
    $D_{ij}^{(l)} \leftarrow \frac{1}{m} t_{ij}^{(l)} + \lambda w_{ij}^{(l)}$
else
    $D_{ij}^{(l)} \leftarrow \frac{1}{m} t_{ij}^{(l)}$
    where $\frac{\delta}{\delta w_{ij}^{(l)}} J(w) = D_{ij}^{(l)}$
```

### 4.4.2 Probabilistic Theory of VAEs

An Autoencoder typically serves as a reconstruction-intended NN since the dimensions of the input and the output layer match and it often behaves in an unsupervised basis. Firstly, the encoder part gradually reduces the dimensionality of the input up to a bottleneck and then increases again until the input dimension is reached (both by using fully connected layers/CNN, or other sort of layers). Conversely, in our thesis, the model architecture has an input dimension of size $[L]$ (1 Frequency Subband with its respective Multipath Components within 1 TTI)

**Algorithm 2** The Adam Algorithm Pseudocode [32]

```
Require: Step size $\epsilon$ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, $\rho_1$ and $\rho_2$ in [0,1). (Suggested
  defaults 0.9 and 0.999 respectively)
Require: Small constant $\delta$ used for numerical stabilization. (Suggested default: $10^{-8}$)
Require: Initial parameters $\theta$
  Initialize 1st and 2nd moment variables $s = 0$, $r = 0$
  Initialize time step $t = 0$
  while stopping criterion not met do
      Sample a minibatch of $m$ examples from the training set $\{x^{(1)}, ..., x^{(m)}\}$ with correspond-
  ing targets $\mathbf{y^{(i)}}$
      Compute gradient: $g \leftarrow \frac{1}{m} \nabla\theta \sum_i L(f(x^{(i)};\theta), y^{(i)})$
      $t \leftarrow t + 1$
      Update biased first moment estimate: $s \leftarrow \rho_1 s + (1 - \rho_1) g$
      Update biased second moment estimate: $r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$
      Correct bias in first moment: $\hat{s} = \frac{r}{1-\rho_1^T}$
      Correct bias in second moment: $\hat{r} = \frac{r}{1-\rho_2^T}$
      Compute update: $\nabla\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$ (operations applied element-wise)
      Apply update: $\theta \leftarrow \theta + \nabla\theta$
```

and an output dimension of $[3 \times L]$ (3 remaining frequency subbands). It also has a supervised approach, aiming to reconstruct a TTI from one observed subband extrapolating to the three unobserved ones. For this reason, we have called it Variational Encoder-Decoder.

According to the formal definition in *Deep Learning* by Goodfellow [34], the encoder can be characterised as the stochastic mapping $p_{encoder}(h \mid x)$ with deterministic function $h = f(x)$, that is the latent representation resulting in the bottleneck . The decoder, in turn, follows with $p_{decoder}(x \mid h)$ and produces a reconstruction $r = g(h)$, that in our case is an extrapolation towards missing information in vector observation $x$. Please see the Figure 4.2 to visualize the main idea of the architecture.

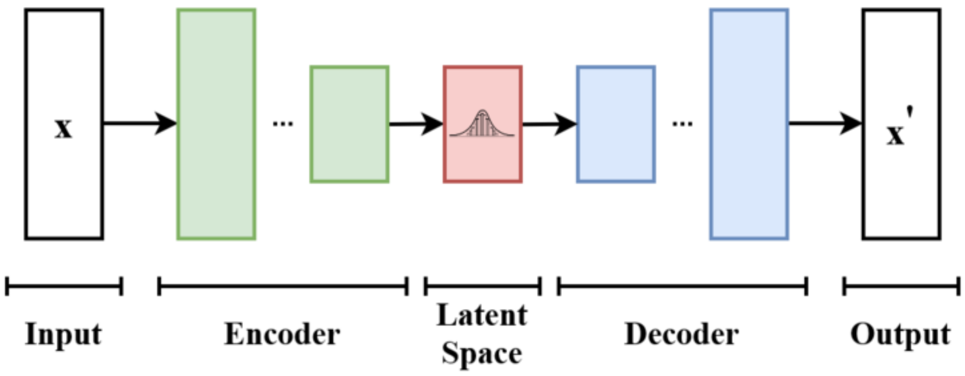


Figure 4.2: Archetypical Variational Autoencoder architecture (obtained from [35])

Leveraging the Variational Encoder-Decoder, we aim at learning the aforementioned distributions from the training dataset. Formally, in Variational Autoencoders (according to [36]), we need to force every point of our training set $X$ ($\vec{x}_i \in X \subset \mathbb{R}^p$) to be mapped to a vector of variables $z$ in a lower-dimensional space $\mathcal{Z}$ that can be sampled from $P(z)$ defined over $\mathcal{Z}$. After, let us consider a family of deterministic reconstructive functions $f(z;\theta)$ parameterized by a vector $\theta$ in a space $\Theta$, where $f : \mathcal{Z} \times \theta \to \mathcal{X}$. $z$ is a random realization whereas $\theta$ is fixed, yielding $f(z;\theta)$ to be a random variable in the space $\mathcal{X}$. The model optimisation intends to tune $\theta$ so as by sampling $z$ from $p(Z)$, $f(z;\theta)$ can reconstruct an approximation of $\mathcal{X}$. In mathematical notation, let $f(z;\theta)$ be replaced by the generative distribution $P(X \mid z;\theta)$. Then, according the law of total probability, the marginal likelihood of the data is to be maximised for each $X$ through the generative process:

$$P(X) = \int P(X \mid z;\theta)P(z)dz. \tag{4.27}$$

The typical output distribution for a VAE is a Gaussian distribution such as $P(X \mid z;\theta) = \mathcal{N}(X \mid f(z;\theta), \sigma^2 I)$ , with hyper-parameter $\sigma^2$. This decision is to increase $P(X)$ via some optimisation technique for some sampled $z$, making the data more likely under the generative model.

In our scenario, we used this intuition but not in a data reconstruction manner. Indeed, we coerced the training into a supervised basis, where the data was arranged by $X_{observed}$ and $Y_{missing}$ tuples, both for the training and validation datasets. As of now, we introduce terms $X_{obs}$ ($X_{observed}$) and $Y_{mis}$ ($Y_{missing}$). That makes explicit that the missing MPCs for each TTI corresponding to the missing frequency subbands are used in this method as regression target values ($Y_{mis}$). Likewise, $X_{obs}$ are used as features/predictors. Having set up this framework, we intended $z$ to capture the interactions between the features of $X_{mis}$ in order for them to generate a proper reconstruction of $Y_{missing}$ by maximising $P(Y_{Missing} \mid z;\theta_{dec})$ updating $\theta_{dec}$ (decoder's parameters). In other words, maximizing the likelihood of the training set under the generative model to being able to reproduce it. It is also crucial to update the encoder's parameters $\theta_{enc}$ in order to learn favourable latent space variables under $P(z \mid X_{obs})$.

#### 4.4.2.1 Mathematical intuition of Variational Autoencoders (VAEs)

As we got inspired by VAEs, in this point we detail the mathematical notion of these generative models that enable the learning of our implemented DL architecture by using Variational Inference (VI) techniques. The assumption in VAE is that $z$ can be drawn from $\mathcal{N}(0, I)$. This is argued by the possibility of reconstructing a dataset $X$ departing from some latent independent normally distributed variables $z$ undergoing a powerful enough function approximator $f(z;\theta)$. The following explanation is based in [36], [29], [37]. In order to maximise Equation 4.27 we need to account only for the probabilities $P(X|z)$ that contribute significantly to estimate $P(X)$. For that reason, a function $P(z \mid X)$ is considered to take a value $X$ and generate a distribution over $z$ values that are more likely to generate a proper reconstructed $X$, and hopefully the space of $z$ is smaller and easier to explore that the one under $P(z)$. To approximate this posterior $P(z \mid X)$ we use a VI framework with $q(z)$:

$$q(z) = \arg\min_q KL(q(z) \mid P(z \mid x)). \tag{4.28}$$

KL is the Kullback-Leibler between $q(z)$ and $P(z \mid x)$, let it be written as:

$$KL(q(z)||p(z|x)) = E_{q(z)}[\log q(z)] - E_{q(z)}[\log p(z,x)] + \log p(x) \tag{4.29}$$

Rearranging Equation 4.29, it can be noted that the KL element is by definition equal or larger than zero, therefore maximising $log\ p(x)$ is to be attained by the Evidence Lower

Bound (ELBO). This is made explicit in Equation 4.30.

$$\begin{aligned}\log p(x) &= KL(q(z)||p(z|x)) - E_{q(z)}[\log q(z)] + E_{q(z)}[\log p(z,x)]\\ &\geqslant -E_{q(z)}[\log q(z)] + E_{q(z)}[\log p(z,x)]\\ &= -KL(q(z)||p(z)) + E_{q(z)}[\log p(x|z)] = ELBO\\ &= E_{q(z)}\left[log\frac{p(x,z)}{q(z)}\right]\end{aligned} \tag{4.30}$$

Note we introduced the distribution $q(z|x) := p(z|x)$ such that it represents the encoder NN, that implicitly models the latent space conditioned on the data, for then reconstructing it by $p(x \mid z)$ as a decoder. Thus, maximising $ELBO$ the VAE training can be performed.

The encoder distribution $p(z \mid x)$ and the prior for latent variables $p(z)$ can be deemed to be Gaussian and approximated by the correspondent NNs. Note that in the following Equation 4.31 $\sigma^2$and $\mu$ depend explicitly on result of the operations between the $x$ fed to the encoder and encoder parameters $\theta$, considered in a normal distribution that yield $z$ by sampling.

$$\begin{aligned}q(z|x) &= \mathcal{N}(z;\mu(x;\theta), I(\sigma^2(x;\theta)))\\ p(z) &= N(z;0,I)\end{aligned} \tag{4.31}$$

#### 4.4.2.2 Reparametrization trick

This setup allows us to perform feed-forward data passes trough the network as we can sample and average $z$ from $q(z \mid x)$ as well as the gradient in ELBO loss. However, we still need to back-propagate the gradient for the weight update in training, which conflicts with $z \sim q(z \mid x)$ and $E_{q(z)}[\log q(z)]$, which is not continuous and lacking of gradient as they come from sampling layers. To work this around, what is performed is the reparametrization trick [29], that consists in embedding the sampling to an input layer, see Figure 4.3. The mean and covariance of $q(z \mid x)$, $\mu(x)$ and $\sigma^2(x)$, are used in combination with another sample $\varepsilon_i \sim \mathcal{N}(0,1)$ to draw our latent variable $z_i = g(x,\varepsilon_i) = \mu_i(x) + \sigma_i^2(x) * \varepsilon_i$. The yielded reparametrized loss expression that is a deterministic function and thus, across a sample of $x_m \in \mathbb{R}^R$ the loss function reads as follows:

$$\mathcal{L}_{\text{VAE}}(\theta) = KL(q(z|x_m)|p(z)) - \frac{1}{L}\sum_{l=1}^{L}\ln p(x_m|z_{l,m}) \tag{4.32}$$

Where $L$ is the number of Monte Carlo samples to average over the term $E_{q(z|x)}[\log p(x|z)]$.

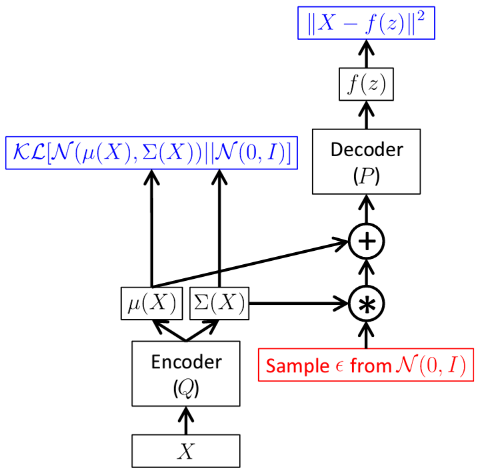
#### 4.4.2.3 Inside the Kullback-Leibler divergence

According to Kullback [38], the KL divergence between to Gaussians is analytically feasible. Consider we have under diagonal covariance Gaussians of dimension $Q$:

$$KL(\mathcal{N}(z_i;\mu_{1,i},\sigma_{1,i}^2)||\mathcal{N}(z_i;\mu_{2,i},\sigma_{2,i}^2)) = \frac{1}{2}\ln\frac{\sigma_{2,i}}{\sigma_{1,i}} + \frac{\sigma_{1,i}^2}{2\sigma_{2,i}^2} + \frac{(\mu_{1,i}-\mu_{2,i})^2}{2\sigma_{2,i}^2} - \frac{1}{2} \tag{4.33}$$

Let the second Gaussian have $\mu_{2,i} = 0, \sigma_{2,i}^2 = 1$ and to abbreviate $\sigma_i^2 := \sigma_{1,i}^2, \mu_i^2 := \mu_{1,i}^2$. Then we yield the loss term entailing the KL divergence:

$$\text{KL}(p(z_i|x)|p(z_i)) = -\frac{1}{2}\ln\sigma_i + \frac{1}{2}\sigma_i^2 + \frac{1}{2}\mu_i^2 - \frac{1}{2} \tag{4.34}$$



$\|X - f(z)\|^2$
$f(z)$
Decoder ($P$)
$\mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) || \mathcal{N}(0, I)]$
$\mu(X)$
$\Sigma(X)$
Encoder ($Q$)
$X$
Sample $\epsilon$ from $\mathcal{N}(0, I)$

Figure 4.3: Variational autoencoder with parametrization trick (obtained from [36])

After that, we need the negative log-likelihood, the so-called reconstruction loss $-\log p(x|z) = -\Sigma_{i=1}^{R} log\, p(x_i|z)$. Let the mentioned R be the output dimension of the decoder, that we deem Gaussian as per assumption $p(x_i|z) = \mathcal{N}(x_i; \mu_i(z), \sigma^2(z))$. Mind that $\mu_i(z)$ is the output of the decoded NN. In practise, what it is forced to be outputted by the decoder instead of $\sigma^2$ is $l_i(x) := \ln \sigma^2(x)$ to force $\sigma^2(x) = exp(l_i(x))$ to be positive. Following the logic of a continuous Gaussian output, the log-likelihood leads to the sum-of-squared error, leveraged as reconstruction loss, analogue to 4.26.

# 5 Experimental Set-up

This chapter is meant to be a thorough explanation on how the experiments with our already explained models were empirically developed (Sections 5.2 and 5.3). Prior to that, in Section 5.1 we describe the software used for each model employed in the thesis.

## 5.1 Software

### 5.1.1 Software - PPCA/BPCA

The programming language used to perform the PPCA's set of experiments is `R` [39], known for its wide-spread usage in statistical and probabilistic modeling. The main library used in these section is `pcaMethods`, leveraged to extract the PPCs and BPCs of our provided datasets. We also used the library `MASS` [40] for some statistical modeling. The fitting algorithms for both BPCA and PPCA implemented in `pcaMethods` are iterative and grow $O(n^3)$ due to the matrix inversions involved.

### 5.1.2 Software - VED

The programming language used to perform the VED's set of experiments is `Python` [41]. The main libraries used in these section are `Keras` [42] and `TensorFlow` [43], in order to build the VED architecture. We also used `Scikit-Learn` [44], `Pandas` [45] and `Numpy` [46] to transform and normalize the data. The plots in these experiments where provided by the package `Matplotlib` [47]. The Scipy.Stats module [48] was used in order to apply statistical modeling to the error of prediction.

## 5.2 PPCA - Experiments

In this section we describe the steps carried out for the PPCA training and assessment.

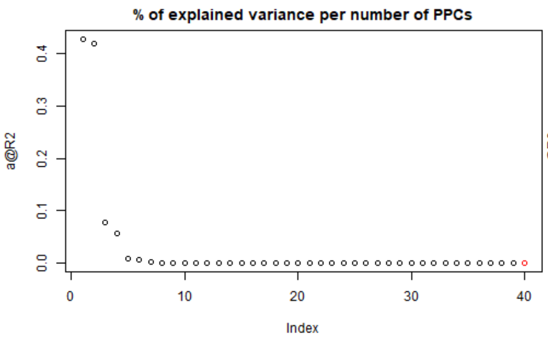
### 5.2.1 Number of PPCs choice

The function `ppca`, was used in the training split of the different datasets in order to obtain a set of values: `loadings` ($W$, the probabilistic principal components), `scores` ($Z$), `center`

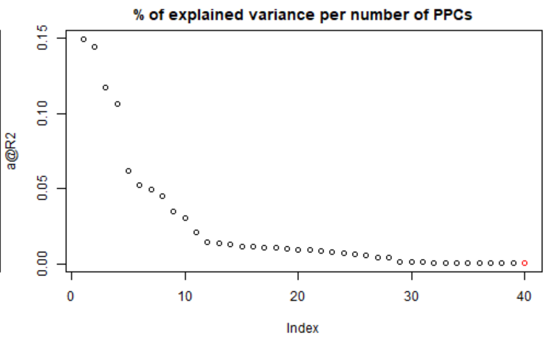
($\mu$, the historic mean of every feature), `R2` (% of explained variance), `R2cum` (% of cumulative explained variance).

Afterwards, a plot of the `R2` was performed in order to decide the optimal number of PPCs ( denoted by `nPPCS`). In this datasets, the maximum computation supported was 40 PPCs, which is deemed as enough because in all cases the % of accumulated variance stagnated from some prior point onwards.

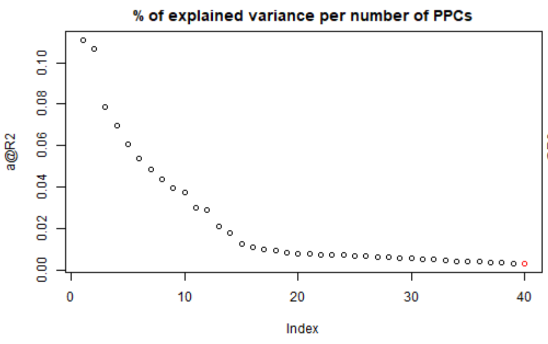
Looking at the graphs in Figure 5.1, we used the heuristics of picking out the `nPPCS` that is prior to the stagnating. There, adding more complex to the model seems not explain more variance, thus leading to possible overfitting.



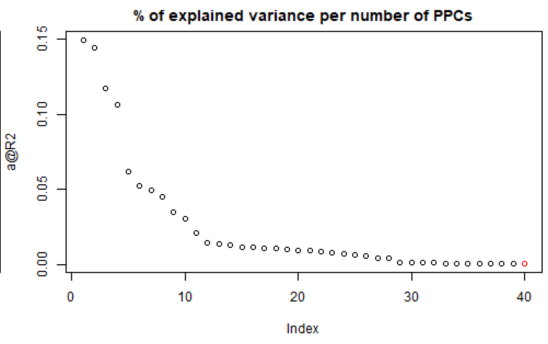
(a) UE0



(b) UE1



(c) UE2



(d) JointDS

Figure 5.1: % of explained variance per number of PPCs (Own elaboration).

Therefore, the number of PPCs used in the model for each dataset are these: `nPPC(UE0)=8`, `nPPC(UE1)=10` , `nPPC(UE2)=15` , `nPPC(JointDS)=12`.

### 5.2.2 PPCA Extraction

Once the optimal `nPPCs` is chosen, we performed the same parameter extraction as in Section 5.2.1. If we want to apply the predictive function of Figure 4.11 we still miss the $I\sigma^2$ term. We estimated this term by performing $\hat{I\sigma^2} = I \cdot VAR(\epsilon) = I \cdot VAR(VEC(WZ^T - X_{train}))$ where $VEC()$ stands for the function that flattens a matrix into a vector.

### 5.2.3 Prediction - PPCA

Leveraging the predictive function in Equation 4.11, we predicted inside each TTI of the validation set. We used the information available in $X_{obs}$ to extrapolate on the $X_{missing}$ MPCs. The results are available in the PPCA Results Section 6.2. In order to take $W_{mis}$ and

$W_{obs}$, we extracted the rows in $W$ that corresponded to the *mis* or *obs* indexes in each TTI. An Analogue procedure was carried out for vectors $\mu_{mis}$ and $\mu_{obs}$.

We also provided Confidence Intervals on this prediction matter by using the predictive function for the covariance in Equation 4.12 and applying the empirical rule $\hat{y}_i \pm 2\sqrt{\hat{\Sigma}_i}$ (where $\hat{y}_i$ is the multivariate prediction for the TTI/observation $i$ ).

#### 5.2.3.1 Cholesky decomposition

In the prediction step, the inversion matrix uses Cholesky decomposition [49]. This is done to deal with the huge dimensions of the covariance matrix to be inverted, which is also highly correlated. This is implemented for the expressions in Equations 4.11 and 4.12, in the form `chol2inv(chol(A))`. This bit of code calculates the inverse of a symmetric, positive definite square matrix from its Cholesky decomposition. These technique follows the reasoning in Equations 5.1, 5.2, 5.3. Let $A$ be the matrix to invert whereas $LL^T = A$ represents the Cholesky decomposition. Note that in Equation 5.1, L is a lower-triangular matrix, this implies that calculating its inverse is faster and more numerically/stable that using the in-built `solve()` function of R.

$$\begin{aligned} Chol(A) = L^T \Rightarrow A = LL^T \\ LL^T x = I \end{aligned} \tag{5.1}$$

Let us introduce the intermediate variable $v := L^T x$. The main idea of inversion by decomposition is the performed in two steps presented in Equation 5.2 and 5.3:

$$Lv = I \tag{5.2}$$

Let us first solve for $v$ and then do:

$$Chol2inv(Chol(X)) \Rightarrow L^T x = v. \tag{5.3}$$

Finally, solving for $x$ in 5.3 we obtain $x = A^{-1}$.

### 5.2.4 BPCA - Experiments

The processes carried out in the BPCA experiments where analogue to the ones in the PPCA. We used the function `bpca` instead of `ppca` and the Bayesian Principal Components are extracted in an analogue way as described in Section 5.2. We used of the number of components chosen in Selection 5.2.

### 5.2.5 Assessment - PPCA

The metrics used to assess the regression performance in the validation set are the following:

$$MSE = \frac{1}{D}\sum_{i=1}^{D}(x_{mis} - \hat{x}_{mis})^2 \tag{5.4}$$

$$MAE = \frac{1}{D}\sum_{i=1}^{D}|x_{mis} - \hat{x}_{mis}| \tag{5.5}$$

$$NRMSE = \sqrt{\frac{1}{D}\Sigma_{i=1}^{D}\frac{(x_{mis_i} - \hat{x}_{mis_i})^2}{mean(x_{mis}^2)}} \tag{5.6}$$

Qualitative plots of the fit of this model are also provided are also provided in Section 6.2.

## 5.3 VED - Experiments

### 5.3.1 Architecture Design

Our VED architectural design is based in the one described in the `Keras` documentation [50] and the already mentioned work of [27]. We opted for the concatenation of a encoder-decoder whose architecture varies for each dataset. We display the architecture choice for Dataset UE0, described in Tables 5.1 and 5.2. The rest architecture parameters for the other datasets are in the Section A of the Appendix. In this tables the "*Output Shape*" column means the number of hidden units for the layer, whereas "*Param #*" corresponds to the number of parameters to be trained in that layer (# Parameters = [# input weights of the layer + 1 bias weight] $\times$ # hidden units in the output).

The encoder and decoder were concatenated with a sampling layer in-between that has been coded as in Listing 5.1. This sampling layer is the one that provides the probabilistic/-variational inference quality to the Encoder-Decoder. Note that in this bit of code we express the property explained in Section 4.4.2.3 of forcing $log\, \sigma^2$. In general, this sampling layer yields a sample from a Normal distribution of [NxM]. $N$ stands for latent dimensions and $M$ for the dimensions provided by the batch size. This is possible by means of the parametrization trick. Please see Figure 5.2 where the diagram of our architecture is visually shown.

#### 5.3.1.1 Heuristics in the Architectural Design

As said, we departed from the design in [27] to base our architecture in some optimal design decision that has been proved to work well in the literature. However, it was necessary to make changes on the architecture because the NN was not learning properly. The number of dense layers was reduced backing this decision up by the loss plots during the training phase (shown in Section 6.3), because the more complex the architecture was getting, the more unstable the learning became. The latent space dimension was also tweaked heuristically according to the same criteria, stable and convergent learning.

```
class Sampling(layers.Layer):
    """Uses (z_mean, z_log_var) to sample z,
    the vector encoding a digit."""

    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon
```

Listing 5.1: Sampling Layer class

### 5.3.2 Training Customisation

For the total loss, we incorporated the KL-divergence in addition to the mean-squared error obtained by comparing $X$ vs $Y$ of every batch. We used implemented Early Stopping as a

Model: Encoder

| Layer (type) | Output Shape | Param | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 618) | 0 | [ ] |
| dense_1 (Dense) | (None, 309) | 191,271 | ['input_1[0][0]'] |
| z_mean (Dense) | (None, 100 ) | 31,000 | ['dense_1[0][0]'] |
| z_log_var (Dense) | (None, 100 ) | 31,000 | ['dense_1 [0][0]'] |
| sampling_1 (Sampling) | (None, 100 ) | 0 | ['z_mean[0][0]', 'z_log_var[0][0]'] |

Total params: 253,271
Trainable params: 253,271
Non-trainable params: 0

Table 5.1: Model summary for the Encoder NN (UE0)

Model: Decoder

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 100) | 0 |
| dense_2 (Dense) | (None, 50) | 5,050 |
| dense_3(Dense) | (None, 1,854) | 94,554 |

Total params: 99,604
Trainable params: 99,604
Non-trainable params: 0

Table 5.2: Model summary for Decoder NN (UE0)

callback, although history plots of both the validation set loss and training set loss along the training for all epochs are provided. It would be useful for a testing subset assessment nevertheless.

The Batch Sizes for the VED vary accordingly to each dataset experiment heuristically. Again, the criteria was to ensure a smooth training that converged. We opted for Adam (explained in Section 4.4.1.3) as an adaptive stochastic gradient descent algorithm, and ReLU as an activation function (remember Equation 4.20) . We implemented as well a hyperparameter to control the weight in the KL-divergence to the contribution in the total loss. We make it explicit it in Equation 5.7 as $\beta$ but we finally set it to $\beta = 1$ for all experiments as the training converged for both losses (MSE and KL) eventually.

$$\mathcal{L}_{\text{VAE}}(\theta) = \beta KL(q(z|x_m)|p(z)) - \frac{1}{L}\sum_{l=1}^{L} \ln p(x_m|z_{l,m}) \tag{5.7}$$

## 5.3.3 Assessment - VED

For the assessment of the VED execution, we relied on the metrics presented in Section 5.2.5 so as we can fully compare model executions in the very same validation set. We also provide in Section 6.3 different plots akin to the ones related to the PPCA experiments.

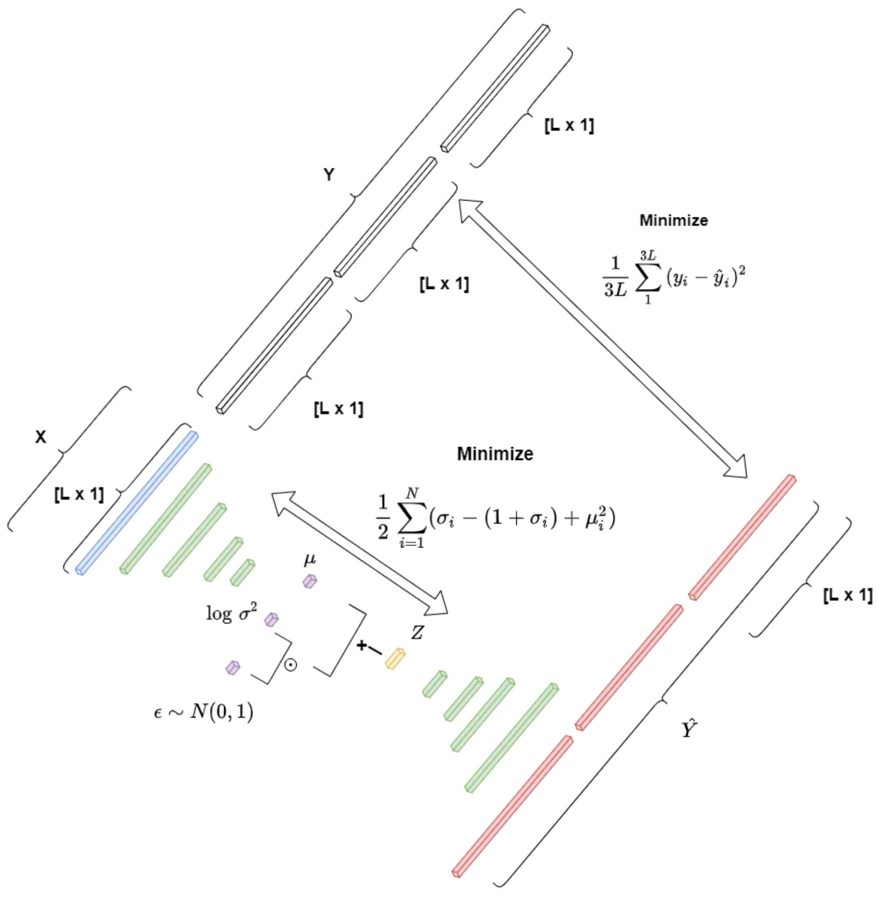


Figure 5.2: Architecture designed for the VED used in the thesis.
The loss function to minimize corresponds to the KL-divergence and the one in the top-right corresponds to the MSE over the predictions against their background truth (Own elaboration).

# 6 Results and Discussion

This chapter presents the results of the PPCA, BPCA and VED models. It also contains further elaborations on these results. Herein we highlight the results that stand out further relating the outcome obtained with the underlying theory. We also argue what could have been wrong with undesirable outputs that were not expected. Moreover, in Section 6.1 we explain the assessment and exploration performed on the data that justifies some of the experimental/design decisions.

## 6.1 Preliminary Statistical Exploration of the Data

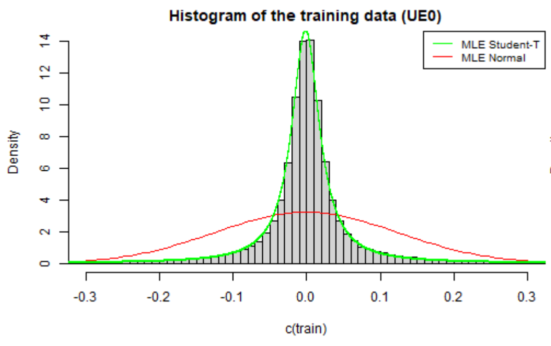
We inspected the data before carrying out the main experiments by using the `MASS R` package. We leveraged the functions **fitdistr** to tentatively fit Normal and Student-t distributions curves in order to know how the data could be roughly modeled. This function uses the Maximum Likelihood Estimation (MLE). The following subsections describe the result of this modeling to further obtain preliminary notions of how the data is distributed.

### 6.1.1 MLE Modeling on the Unmasked Training Set

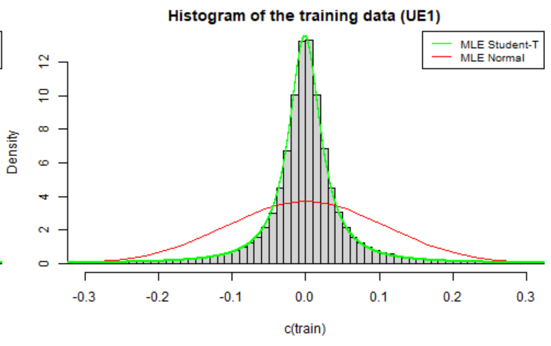
We applied this MLE modeling to the unmasked training splits of the datasets. The curves of the Gaussian and the Student-t distributions are observable in Figures 6.1a and 6.1b for UE0 and UE1 datasets. The data in the histogram is clearly reflected not to be Gaussian (so was this reflection in the remaining datasets). As for this modeling, when the Gaussian curve is not able to capture all the data in the tails' support, it attempts to widen itself increasing the variance, hence trying to cover the most deviated data. In this way, it fails at enveloping the whole histogram (it does not envelope the central part where there is most of the density). In the case of the Student-t distribution, by tweaking the parameter of *df* (degrees of freedom that control the Kurtosis) the data gets modeled best. We assume that the rest of the data (the validation set) shares the same distribution as the training data.

### 6.1.2 Modeling the PPCA Reconstruction Error

As we have seen in Section 6.1.1, the data is not normally distributed. We extracted Probabilistic Principal Components (PPCs) and applied the reconstruction of the data through



(a) UE0



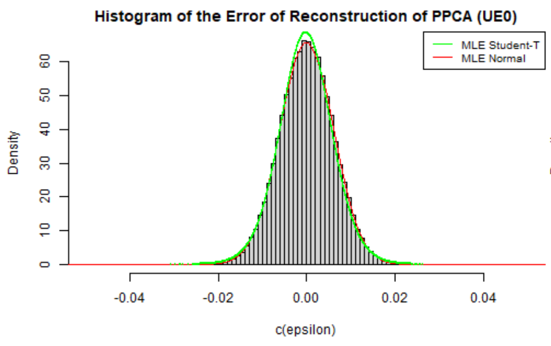
(b) UE1

Figure 6.1: MLE modeling on the training subset (Own elaboration)

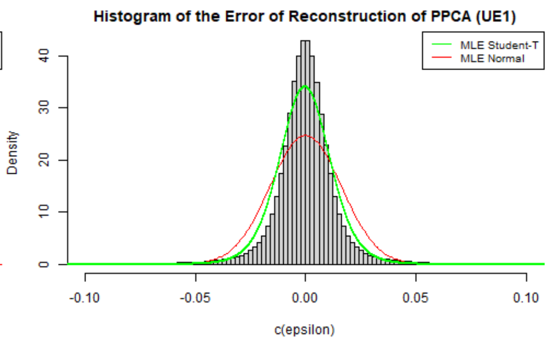
multiplying the obtained $W \times z^T$ (loadings times transposed scores). Then we subtracted the product from the training dataset to get the error: $\epsilon = X_{train} - W \times z^T$, as described in Section 4.1.

Hence, we overlaid the MLE curves on the histogram of this error $\epsilon$. In the case of UE0 (Figure 6.2a), the error is considerably normal distributed, therefore we can assume that the 8 PPCs extracted capture the distribution as the reconstruction yields this $\epsilon \sim \mathcal{N}$. However, more density of probability is spread towards the tails in UE1 train dataset, Figure 6.2b. That reflects the inability of PPCA to preliminary capture the distribution of the data as the error cannot be approximated as Gaussian (the outcome in the remaining datasets resembles most to this UE1 result). We could tentatively address this problem to the fact that the CSI of some TTIs would not belong to the same coherence window anymore. It is not deterministically clear what the actual issue could be, yet accounting for the fact of having such thick tails we could consider the error falls more apart from $\mu = 0$ because of the high variability in CSI over time.

As for UE1, we intuited that the PPCA predictive capability would not fare as well as in UE0 dataset due to the non-gaussianity of its reconstruction error $\epsilon$ and the intrinsic non-gaussianity of the data. We observed a similar behaviour in the modeling between UE1 and UE2.



(a) UE0, 8 PPCs



(b) UE1, 40 PPCs

Figure 6.2: MLE modeling on the PPCA reconstruction error, training subset (Own elaboration)

### 6.1.3 Estimated Covariance Heatmap

Let us consider $WW^T$ as an approximation of the covariance $\Sigma$ taken out of UE0 dataset, obviating the term $\sigma^2 I$. Note that $W$ is the factor loadings extracted from the PPCA fitting process, remind Equation 4.2. We made use of the library `seriation` [51] to obtain a heatmap of this covariance matrix. Mind the truncation trick used in Listing 6.1 to augment the resolution of the heatmap of Figure 6.3.

In this Figure, every pixel represents the correlation between each of the 2472 of the effective unfolded UE0 dataset. We can observe there exists a pattern of squares. Each square in the diagonal represents the correlations within either the real or imaginary features belonging to one Frequency SB. The off-diagonal squares represents correlations of either real or imaginary features against their conjugate counterpart in the same subband or against the real or imaginary features of other subbands. In Figure 6.4, two closeups of the Figure 6.3's heatmap are shown. The right heatmap is a slice of the correlations of two Frequency SBs, whereas the left image corresponds to the real features of the first $SB$. At square level, the higher correlations are in the diagonal (auto-correlation) and in the top-left section, meaning that there is a stronger correlation between the first features (MPCs) of each Frequency SB (on its own) and between the first MPCs belonging to different Frequency SBs.

```
S <- pc_loadings %*% t(pc_loadings)
max = 2 * sd(S)
S[which(S>=max)] = max
S[which(S<=-max)] = -max
```

Listing 6.1: Truncation of $WW^T$ values, (R Code)

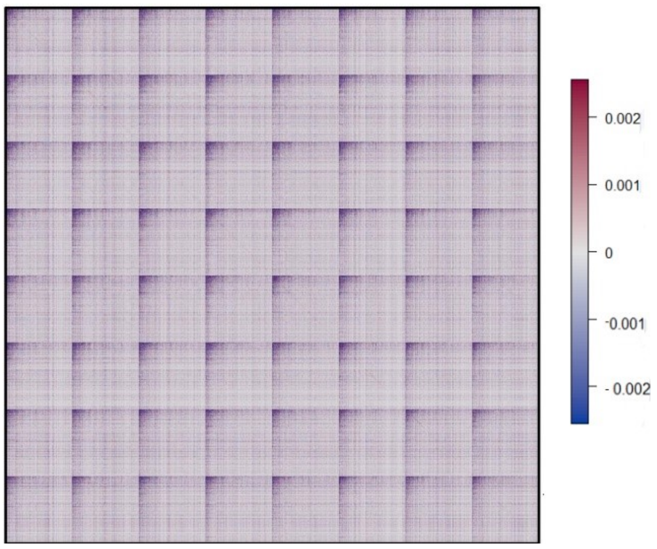


Figure 6.3: Heatmap of the $WW^T$ , [2472 x 2472] (Own elaboration)

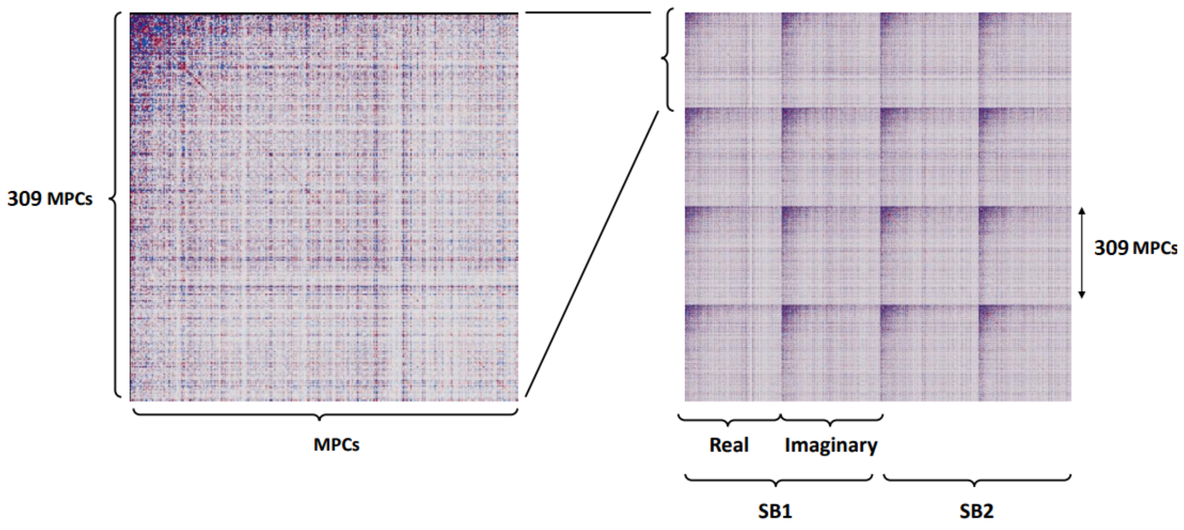


Figure 6.4: Close-up of Approximated Covariance Heatmap (Own elaboration)
Left, [309 x309] heatmap of the Real part of the MPCs belonging to the 1st Frequency SB.
Right, correlations of the features belonging to the 2 first SBs, Real and Imaginary parts highlighted.

## 6.2 Results - PPCA & BPCA

By following the described prediction algorithm in Section 5.2, we show in Table 6.1 the results of the prediction in the validation set of each dataset employed. As a visual support of these prediction outcomes, we attach several plots that reflect how the model executions developed.

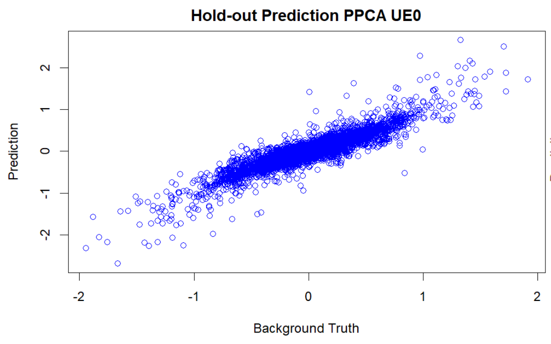
Table 6.1: Error Metrics of PPCA/BPCA Experiments

| | | Dataset | | | |
|---|---|---|---|---|---|
| Experiment | Error Metric | UE0 | UE1 | UE2 | JointDS |
| PPCA | MAE | **0.0276** | **0.0513** | **0.0720** | **0.0664** |
| | RMSE | **0.0545** | **0.1062** | **0.2417** | 0.1573 |
| | MSE | **0.0029** | **0.0112** | **0.0584** | 0.0247 |
| BPCA | MAE | 0.0276 | 0.0514 | 0.0996 | 0.0665 |
| | RMSE | 0.0545 | 0.1064 | 0.5892 | **0.1568** |
| | MSE | 0.0029 | 0.0113 | 0.3471 | **0.0245** |
| nPPCs | | 8 | 10 | 15 | 12 |

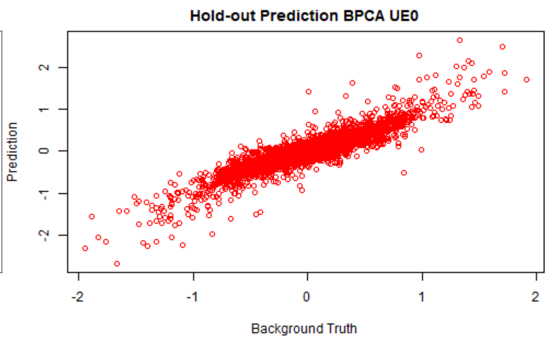
In Figure 6.5, it is shown how the predictions are scattered versus the corresponding background truth values for the PPCA and BPCA learnt models in all of the four datasets. Thus, the thinner the cloud of points, the better the prediction. Likewise, a slope close to 45º with respect to the horizontal implies a better overall prediction.

Moreover, in Figure 6.6 we show how the predictions are visually presented over all the features across one time sample (one TTI) for the PPCA model in datasets UE0 and UE2. In Subfigure 6.6a we can see that the image presents considerably less distant predicted values (red) to the background truth than in Subfigure 6.6b. This is reflected in the results of Table 6.1 if extrapolated for all the dataset. Note also that there are black dots in Subfigure 6.6b quite deviated from the main cloud of points.

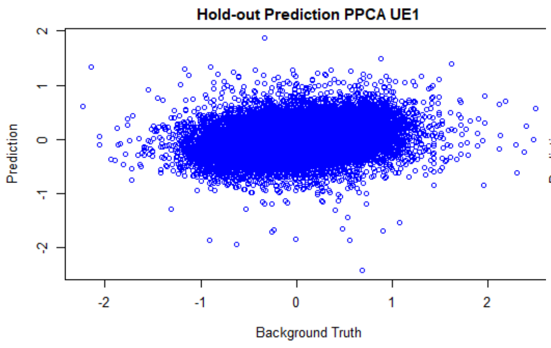
To visually asses how the predictions behave over the intended support, we sorted all the background truth values of the validation subset by amplitude and overlaid the corresponding predictions (see Figure 6.7). In this figure we show the sorted predictions of datasets UE0 and



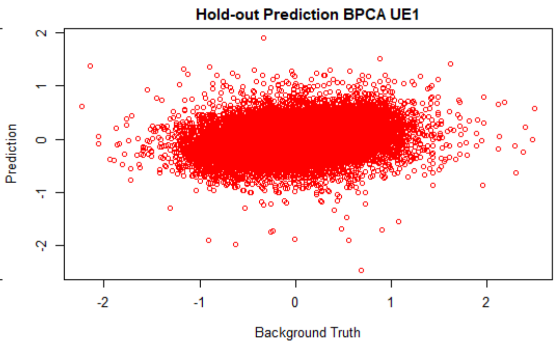
(a) UE0, PPCA



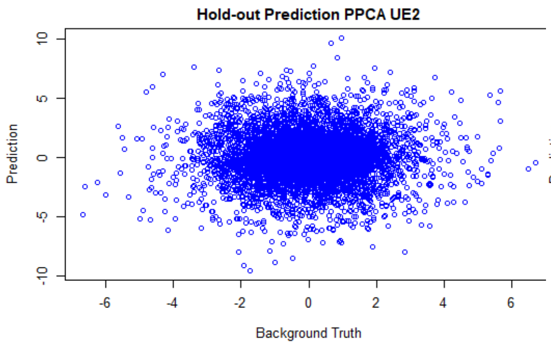
(b) UE0, BPCA



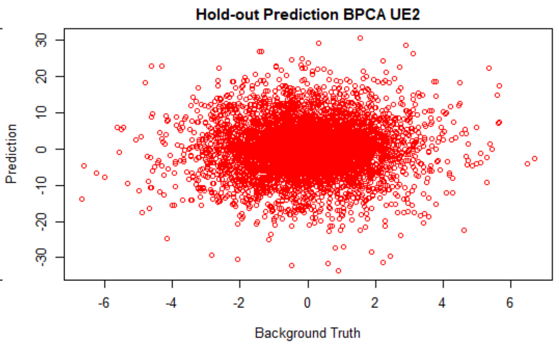
(c) UE1, 29 PPCs



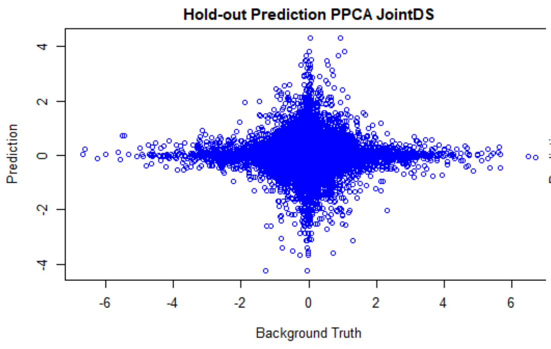
(d) UE2, 15 PPCs



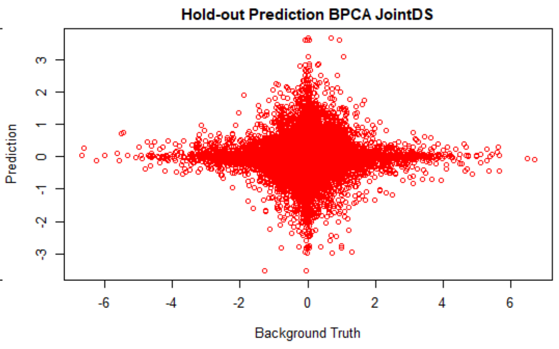
(e) UE0, 8 PPCs



(f) JointDS, 12 PPCs

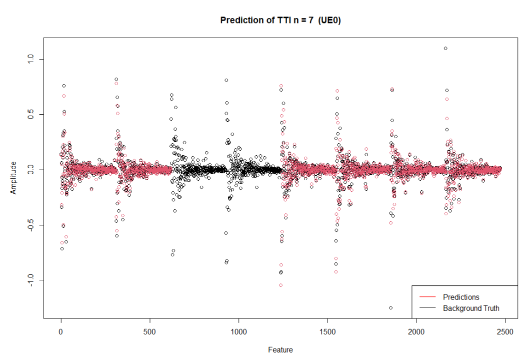


(g) UE0, 8 PPCs

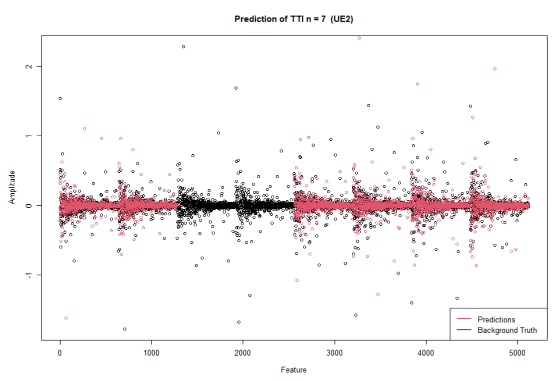


(h) JointDS, 12 PPCs

Figure 6.5: Validation subset prediction plot for the missing values (PPCA) (Own elaboration). $\hat{y}$ On the vertical axis versus $y_{true}$ in the horizontal



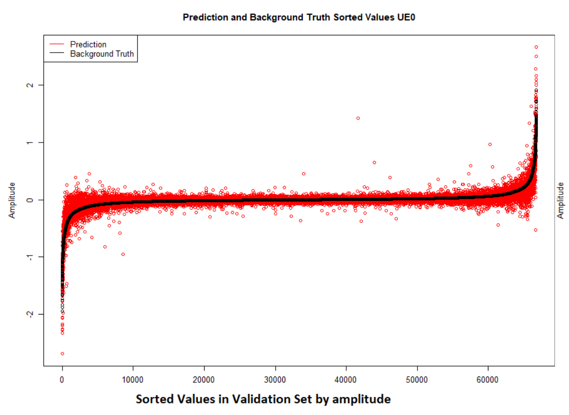
(a) UE0



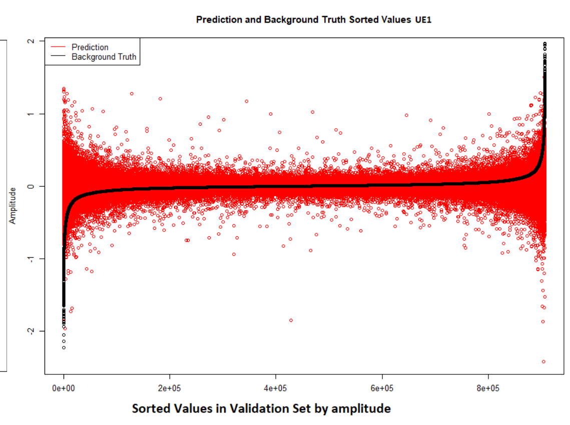
(b) UE2

Figure 6.6: Plot of the Amplitudes for one TTI. PPCA - UE0, UE2 (Own elaboration). Note the black dots represent the actual observed Multipath Components, whereas the red correspond to the predictions. The segment of black values with no overlaid red is the observed Frequency Subband.

UE1. Note that the predictions of UE0's validation split seem to capture more properly the range of values. UE2 and JointDS datasets incur the same behaviour as the shown UE1 in Figure 6.7b, see Appendix B for related figures.



(a) UE0

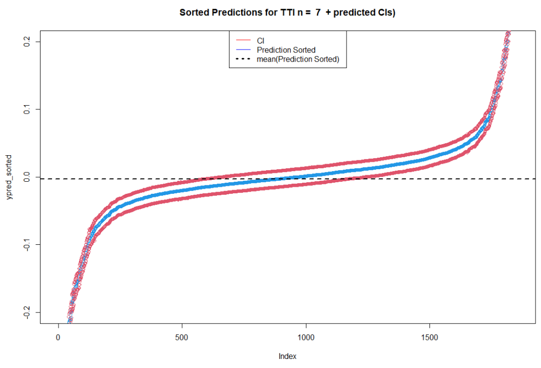


(b) UE1

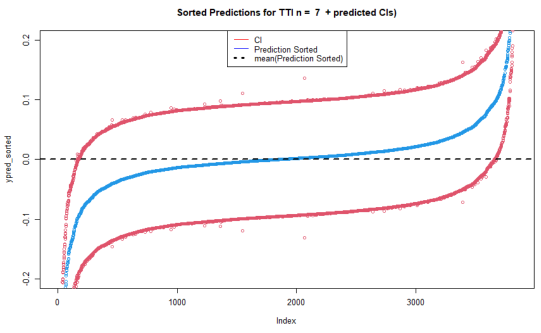
Figure 6.7: Plot of the Amplitudes for all Predictions. PPCA - UE0, UE1 (Own elaboration). Note the black dots represent the actual observed Multipath Components (Re o Im part), whereas the red correspond to the predictions.

We also wanted to assess some measures of uncertainty in the predictions. Therefore, we display in the Figure 6.8 how the confidence intervals look in the predicted, sorted values of one single TTI. There, it is obvious that in the case of the UE0 dataset, the predictions are more certain than in UE1. Taking a closer look to Subfigure 6.8a, we observe wider CI bands than in Subfigure 6.8a. Both Subfigures, though, present more uncertainty the closer they are from their mean, which is near to the 0 amplitude value. The fact that the prediction of the model is more certain in UE0 matches what is already explained about Figure 6.6, less distant values compared to the background truth. See more related Figures for the remaining datasets in Appendix B.

Finally, we took the predicted errors for each dataset's validation subset in a 1:1 fashion and flattened them into their respective 1-dimensional error vectors. Later on, we applied



(a) UE0



(b) UE1

Figure 6.8: Confidence Intervals plot in the prediction of a TTI. PPCA - UE0, UE1. (Own elaboration)

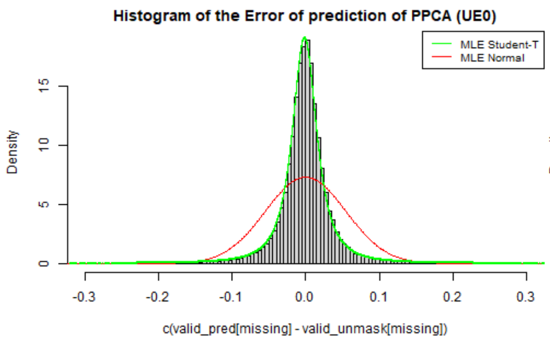
These plot shows the sorted redicted values for 1 TTI along with the Confidence Intervals overlaid.

the same MLE modeling we performed in Sections 6.1.1 and 6.1.2 to these flattened vectors. We show these modeling outputs in Figure 6.9. Note the modeled red MLE-fitted Gaussian curve in Subfigure 6.9a is the one that seems to be capturing the data in the histogram the most. In overall, all the plots in all these Subfigures look like Student-T-distributed rather than Gaussian distributed. This could be alleged to the inability of PPCA to capture the correlations between the features by linear-gaussian modeling.

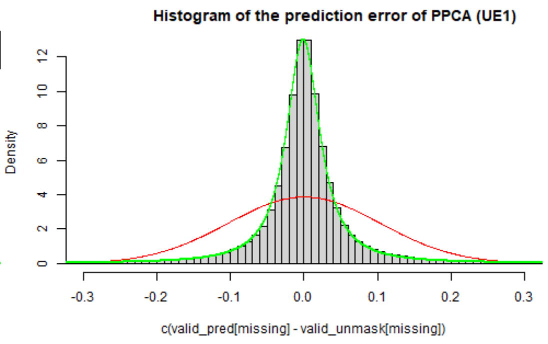
### 6.2.1 Discussion on the PPCA and BPCA Results

Once the model was fitted and we further carried out the predictions and the statistical modelings, we could observe that in overall the PPCA model attained better performance than BPCA. The fact that the BPCA slightly underperformed the PPCA could be because the library implementation designed by Oba [23] is intended for missing value estimation rather than a hold-out training/prediction approach. As orthogonality is not forced in the fitting of the factor loadings, that may suppose that the new coordinate system defined for them does not capture the greatest variance for the scalar projections of the data as optimally as PPCA does. This method is alleged to perform better when the missing values are embedded in the datapoints that make up the training set where the EM fitting is performed. That is to say, in a value imputation approach. However, as seen in Figure 6.5, both methods present very similar predictive behaviours.

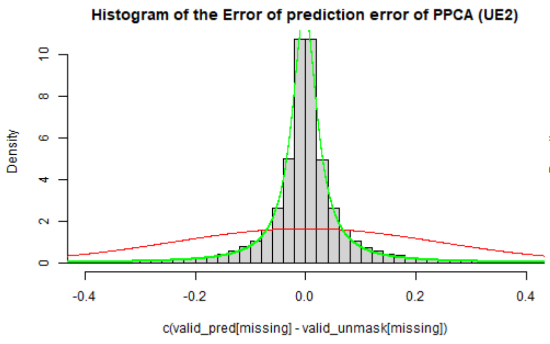
According to Figure 6.5, the best prediction is undoubtedly the one in UE0 dataset, see Subfigures 6.5a and 6.5b. It can be argued that in that dataset the predictions are more accurate (mind also that in Table 6.1 it presents the best results) due to the scarcity of TTIs



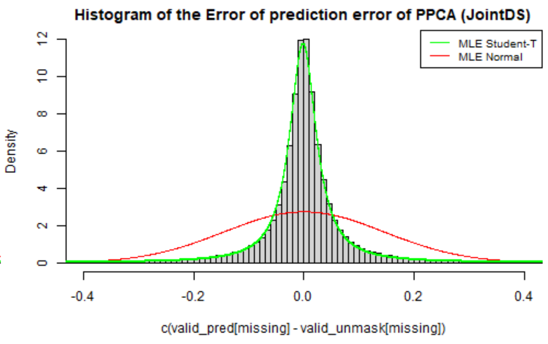
(a) UE1, 29 PPCs



(b) UE2, 15 PPCs



(c) UE0, 8 PPCs



(d) JointDS, 12 PPCs

Figure 6.9: MLE modeling on the PPCA prediction error, validation subset (Own source).

in comparison to the other datasets. Remember that the variation in the Channel becomes less linear and smooth the more we get far in time from the Coherence Window. As this interval is suppose to be around 80 TTI in the simulated datasets, we expect an overall decent prediction in this dataset.

The distributions of the predictions of UE1 and UE2 (Subfigures 6.5c to 6.5f) do not show a cloud of points with such a good inclination. They also reflect thicker clouds of points. We could map the thickness of the clouds to the variance/uncertainty or instability in the predictions, since for similar background truth values the model is predicting a remarkably different range of values. The inclination is more intrinsically related to how good in average the model is able to predict properly for the whole support of values. Then, having a horizontal cloud of points means a poor fit of the model, as it is in average predicting the same for all the support of background truth values. We could blame this drastic deterioration on the prediction to more pronounced changes in the Channel over time, due to the large quantity of TTIs in comparison to the Coherence Window.

In the case of JointDS, Subfigures 6.5g and 6.5h, the prediction distribution seems to reflect a bivariate Student-T distribution. The fact this dataset is a merge out of UE1 and UE2 could have yielded some counterproductive correlations in the process of covariance approximation of PPCA. In this Subfigures it can be clearly observed that the prediction is completely random for the amplitude mean of the background truth data (close to 0). Conversely, it predicts 0 for the rest of the support of background truth values.

In general, the less Gaussian-distributed was the reconstruction error of the datasets (see plots in Figure 6.2) the better the prediction was. We could relate this property to the Gaussian assumption of the PPCA/BPCA models in the data and the error. See how in Figure 6.9 the error is mostly Student-T-distributed instead of Gaussian. In the case of UE0 (Subfigure

6.9a) the Gaussian curve attains to fit the data better than in the rest of the cases. This phenomenon reflects how the Gaussian assumption prevents the model to capture the data under the density of probability in the tails, which is large.

This uncertainty in the prediction in UE1, UE2 and JointDS datasets is also reflected in SubFigure 6.8b, which in comparison to the uncertainty reflected in UE0 (Subfigure 6.8a) is much larger for most of the values, visible at the width of the Confidence Intervals. However, the plots in Figure 6.7 reflect how for UE1 (b), the values that go away from the mean are not captured (it happens the same for UE2 and JointDS, see Appendix B). The model just increases the variance for those amplitudes in that range, failing to output predictions with the correct sign. It is not the case for UE0 (a), where these outliers in the dataset are better captured by the PPCA model. This is also reflected in Figure 6.6, where for UE2, it is clearly visible that beyond some threshold, the predictions fail at capturing the datapoints. In UE0 (b), conversely, we see that PPCA is closer to capture these points. The property of PPCA of departing from the historical mean of the target feature and then leveraging the predictive conditioned covariance could be a explanation for the goodness of the predictions in UE0.

We should also point out that, besides the increase in regards to TTIs of UE1, UE2 and JointDS with respect to UE0, there also exist an increase in the feature space, from 2472 effective features to 5120. Our practical intuition leads us to think that having ten times more datapoints than features might have also helped the model to perform a better learning of the underlying correlations. It is also remarkable that having to predict for 75% of the values at each datapoint has been such a determinant limiting factor for the aim of learning a reliable regression model.

## 6.3 Results - VED

After having trained the models and further predicted for the validation splits' values, we show in Table 6.2 the results of such experiments for each dataset employed. As a visual support of these prediction outputs, we attach several plots that reflect how the model executions developed.

Table 6.2: Error Metrics of PPCA/BPCA Experiments

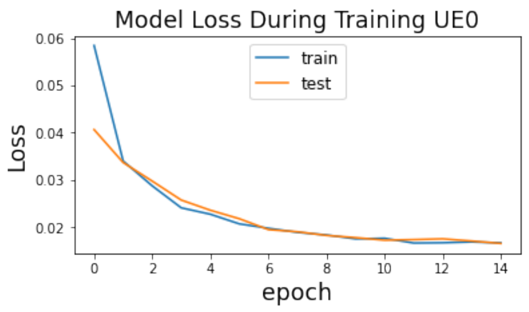
| | Dataset | | | |
|---|---|---|---|---|
| Error Metric | UE0 | UE1 | UE2 | JointDS |
| MAE | **2.1730** | 11.801 | 16.7114 | 28.525 |
| RMSE | **0.7671** | 1.6105 | 2.6820 | 3.1247 |
| MSE | **0.5864** | 2.5927 | 7.2108 | 9.7785 |

The tracked losses in the training stages of the four architectures are arranged in Figure 6.10. There, we can observe that both training and validation losses converge for all the datasets.

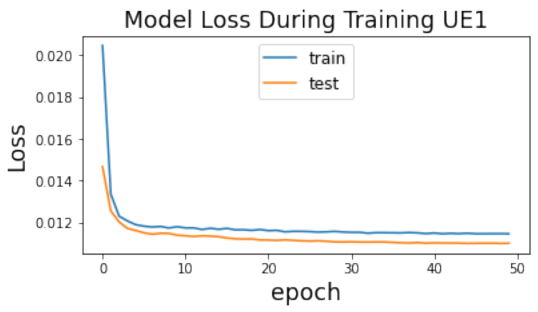
Similar to what we did in the PPCA/BPCA experiments, we visually assessed how the predictions are distributed against their respective background truth values (Figure 6.11).

Again, analogue to the PPCA experiments, in Figure 6.12 we present how, for a single TTI, the predictions are arranged. It can be noted that there is an inability of the model when predicting for values over a threshold. Although, these bigger values are better captured in the UE0 experiment (6.12a).

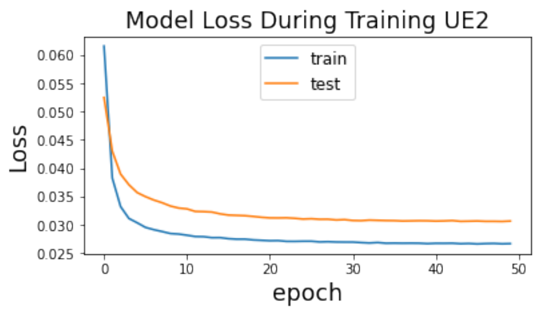
We also performed the plots of all the sorted validation set's predicted values for UE0 and UE1 (Figure 6.13). Similar outcome was produced for the remaining datasets, see Appendix C. It is visible that the models have troubles to capture values larger than some range in positive and negative amplitudes. There also exists a great variance on the values predicted.



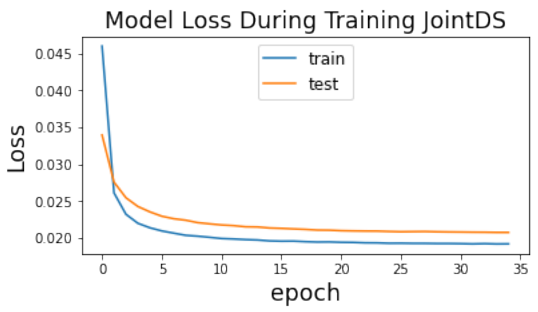
(a) Training Losses UE0



(b) Training Losses UE1

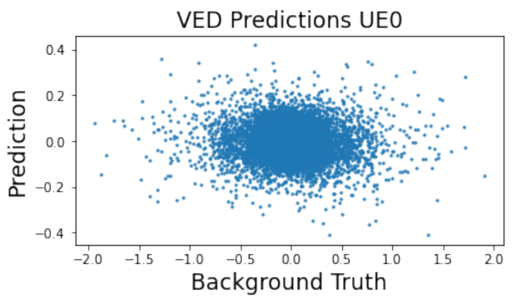


(c) Training Losses UE2

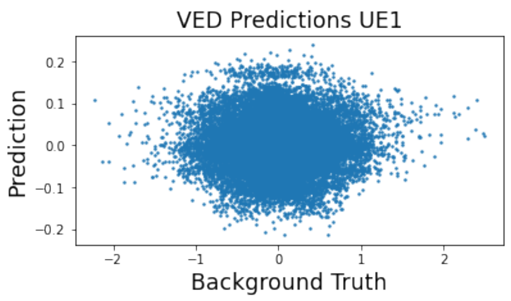


(d) Training Losses JointDS

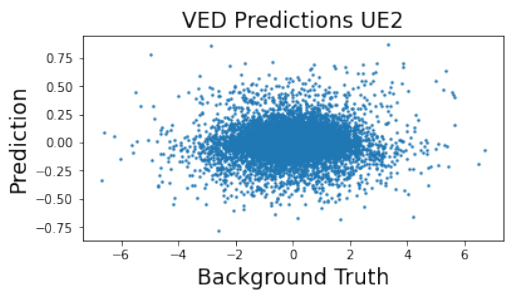
Figure 6.10: Training Losses for the NNs in all datasets (Training and Validation Splits) . (Own elaboration)



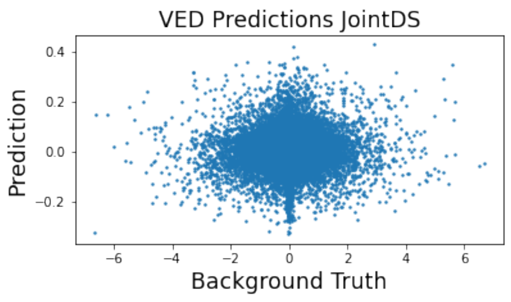
(a) UE0



(b) UE1



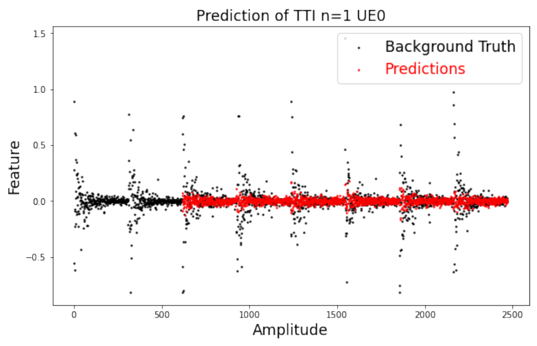
(c) UE2



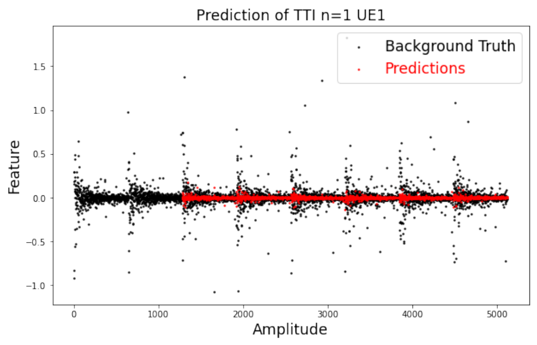
(d) JointDS

Figure 6.11: Validation subset prediction plot for the missing values (VED) (Own elaboration). $\hat{y}$ on the vertical axis versus the $\hat{y}_{true}$ in the horizontal.

The amplitudes unable to be reached are notably less present among the validation set, we could consider them to be outliers, even though the model fails at reaching them.



(a) UE0



(b) UE2

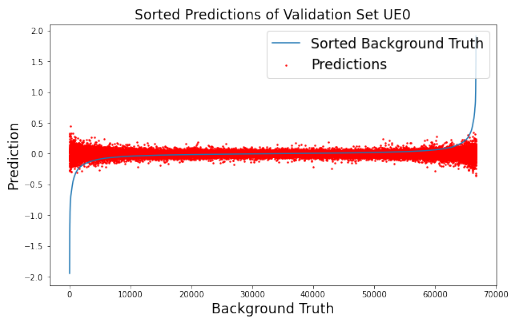
Figure 6.12: Plot of the Amplitudes for one TTI. VED - UE0, UE1 (Own elaboration). Note the black dots represent the actual observed Multipath Components, whereas the red correspond to the predictions. The segment of black values with no overlaid red is the observed Frequency Subband.

Finally, same modeling of the prediction error as in the PPCA/BPCA experiments is performed (this time with `Scipy.Stats Python` library), see Figure 6.14. Again the more similar to a Gaussian distribution looks to be UE0, but all of them are best fitted by a Student-t distribution.

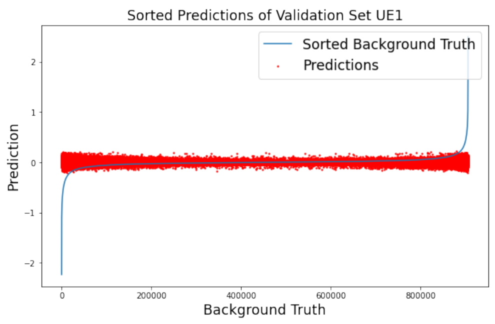
### 6.3.1 Discussion on the VED Results

In this Section we proceed to carry out a general commentary on the results as we did in Section 6.2.1 for the baseline. Generally, the VED models obtained worse performances than the baseline, (around an order of magnitude of $\times 100$ in the case of the MSE). Ensuring a convergent and stable training was something we first struggled with when choosing the architecture hyperparameters (number of layers, type of activation functions, number of hidden units, dimension of latent space, loss function, etc.). Then, we tried to vary heuristically these hyperparameters in order for the output to try to capture the values of larger magnitude. In Figure 6.12 we see that the values beyond some threshold are not predicted properly. This is more accentuated in UE1 6.12b, where the span of prediction is narrower, even (same output for UE2 and JointDS, available in Appendix C).

The mentioned phenomenon is also pronounced in Figure 6.7, where we plotted for UE0 and UE1 the predictions of the validation set sorted by amplitude. As in the PPCA case, the model fails to capture these outliers. Note that we can see this fact reflected in the statistical modeling of the prediction error in Figure 6.14, where it is shown the incapability of capturing these extreme values in the thick tails for all datasets. Again, the Student-T distribution would be a more suitable distribution of this error.

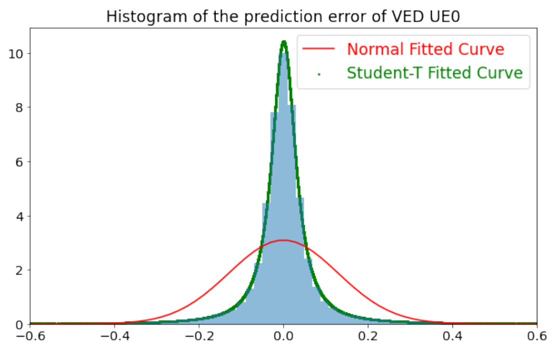


(a) UE0

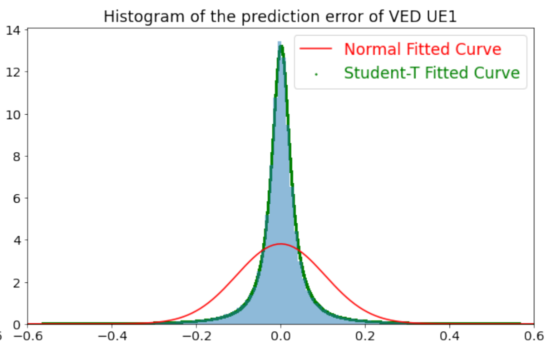


(b) UE1

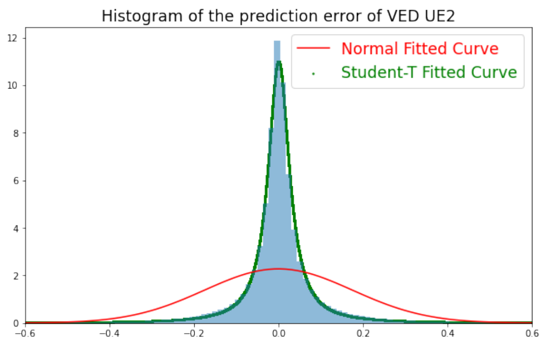
Figure 6.13: Plot of the Amplitudes for all Predictions. VED - UE0, UE1 (Own elaboration). Note the black dots represent the actual observed Multipath Components (Re o Im part), whereas the red correspond to the predictions.



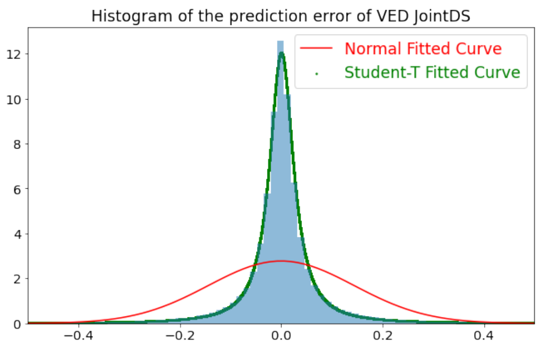
(a) Prediction Error Modeling UE0



(b) Prediction Error Modeling UE1



(c) Prediction Error UE2



(d) Prediction Error JointDS

Figure 6.14: MLE modeling on the VED prediction error, validation subset (Own source).

As a general note on the prediction quality, we presented scatter plots of the predictions versus the background truth values in Figure 6.11. All the clouds of points seem Gaussian-distributed, although the lack inclination and thickness reflect inaccuracy of the predictions.

In overall, it looks that the model has failed in capturing the data distribution, as well as in its main purpose, which was learning the non-linear correlations of the features to increase the quality of the predictions in relation to the baseline. The squared pattern in the covariance heatmap of Figure 6.3 describes a strong correlation of the first MPCs intra/inter-frequency subband. We wanted to exploit this scenario by performing the dimensionality reduction, thinking that this important correlations would be kept and outweighed the irrelevant ones. We addressed the cross-band extrapolation problem by using a variational inference approach and relying in its generative distribution approximation power for a proper feature reconstruction. Generally, the model presents clear signs of overfitting since the output relies too much in predictions close to the training data mean, whose presence in the dataset is overly abundant (high kurtosis Student-T distribution).

A possible downside of the model chosen could have been the Gaussian assumption of the loss function (MSE), that considers the output to be normally-distributed, despite the Statistical modeling performed reflects not to be the case. A likelihood function resulting from the Student-T distribution could be tried as an alternative loss function instead of the mean squared error, aiming to capture extreme values that fall more distant to the mean in the tails of the data distributions for all the datasets. The variation in the hyperparameter that regulates the weigh of the KL loss could also be explored in order to ensure that its contribution does not make the model overfit. As a regularization strategy, dropout [52] in the layers could be also applied, aiming to reduce overfitting.

# 7 Ethical Considerations

This thesis work is commissioned by The RD Center of Huawei Technologies Sweden AB in order to explore the Channel Prediction problem which is encompassed in the wireless communications field. There are several related considerations that could have an impact to society and the environment to some extent. The ultimate aim of this work is to increase the throughput capacity, data-rate, spectral efficiency and network capacity of wireless transmissions in MIMO scenario between users and base stations. To tackle this problem, we isolated the scenario into a SIMO approach, thus focusing in one UE and multiple antennas to try to revamp the communication transmission's quality.

For the safety of the environment and individuals, security policy standards that rule a fair use of the frequency bandwidth have to be followed depending on the technology used, such as 5G. Overlapping an unintended frequency bandwidth could occasion harmful troubles for the safety of other individuals and organizations. That is because this fact could compromise somenone else's communication due to interference. Another issue to comply with is the quality standards for a reliable communication that one has to aim for, such as the *Regulation (EU) 2015/2120 of the European Parliament [...] towards electronic communications networks and services and Regulation (EU)* [53].

# 8 Conclusion

It has been possible to create a method of cross-band channel prediction and further evaluate it with statistical metrics as well as with visual assessment. It turned out that the experiments conducted in UE0 was the most satisfactory one. The most revealing assessment was the visual one we carried out by plotting the predictions along individual TTIs and over all the validation dataset. Through this visual, qualitative evaluation, we could address the good quality of this results to a successful covariance matrix approximation through dimensionality reduction by means of PPCA. This could be explained because all the TTIs might be forming part (approximately) of the same Coherence Window. Thus, linear correlations between MPCs could be leveraged for the prediction of missing ones by using the observed information for each TTI.

In the case of the VED, the results were not so satisfactory, showing worse prediction results. We could address this problem to the overfitting occurred during the training, where the VED was reflected not to have captured the non-linear relations between the features for a proper extrapolation. Therefore, we propose some further work that could circumvent this event. A tweak that could be performed to the VED architecture would be to try a loss function of another nature, like the likelihood function of a Student-t distribution. This would assume the data distribution not to be Gaussian, aiming to capture the outliers more effectively. Likewise, this shift of assumption might be worth to be assessed in the sampling layer, switching to a Student-T distribution in the bottleneck of the VED where the variational inference takes place.

On another note, further regularization strategies should be tried, e.g., dropout. In this line, further hyperparameter searching could be implemented at the level of VED architecture: hidden units and layers, dimension of the latent space, activation function, etc.

On the evaluation matter, the assessment of the prediction's quality could revamped by adding posterior stages of signal reconstruction. This would allow to use intrinsic wireless communications performance metrics such as beamforming gain, spectral efficiency and Signal-to-Noise ratio, that were not use in the thesis due to time and background constraints. These metrics would use the reconstructed $H$ matrices instead of the MPC, allowing to compare the results directly to literature benchmarking and state-of-the-art.

Concerning the inter-TTI prediction purpose, due to the complexity of implementing a framework that combine feature-wise correlations with time-domain ones, it has been an unfeasible goal that we must leave for a future research track. These approach could be worked out by the use of LSTMs as hidden layers in the VED, using different order $p$ of previous TTIs involved in the time correlation in the learning.

Finally, reproducing the experiments with a larger database in terms of TTIs (in the order of tens of thousands) could fare better with the currently implemented models, as it is usually a desirable method to prevent overfitting from happening.

# Bibliography

[1] Robin Chataut and Robert Akl. “Massive MIMO Systems for 5G and beyond Networks—Overview, Recent Trends, Challenges, and Future Research Direction”. en. In: *Sensors* 20.10 (May 2020), p. 2753. ISSN: 1424-8220. DOI: 10.3390/s20102753. URL: https://www.mdpi.com/1424-8220/20/10/2753 (visited on 03/16/2022).

[2] *Understanding Massive MIMO technology.* URL: https://www.avnet.com/wps/portal/abacus/solutions/markets/communications/5g-solutions/understanding-massive-mimo-technology/.

[3] *Huawei’s 5G Indoor Distributed Massive MIMO — "Best Solution Case" by ICT China 2021.* Sept. 2021. URL: https://www.huawei.com/en/news/2021/9/ubiquitous-gigabit-5g-indoor-dmm-wins-award (visited on 02/28/2022).

[4] Francois Rottenberg, Thomas Choi, Peng Luo, and Jianzhong Zhang. “Performance Analysis of Channel Extrapolation in FDD Massive MIMO Systems”. In: (). URL: https://www.researchgate.net/figure/Massive-MIMO-multipath-propagation-environment-described-by-L-components-Each-multipath_fig6_332139039.

[5] *mmWave Massive MIMO.* en. Elsevier, 2017. ISBN: 978-0-12-804418-6. DOI: 10.1016/C2015-0-01250-3. URL: https://linkinghub.elsevier.com/retrieve/pii/C20150012503 (visited on 03/16/2022).

[6] Jubin Jose, Alexei Ashikhmin, Thomas L. Marzetta, and Sriram Vishwanath. “Pilot Contamination and Precoding in Multi-Cell TDD Systems”. In: *arXiv:0901.1703 [cs, math]* (June 2010). arXiv: 0901.1703. URL: http://arxiv.org/abs/0901.1703 (visited on 02/28/2022).

[7] Communication ACM SIGCOMM Conference ACM Special Interest Group on Data, Association for Computing Machinery, Special Interest Group on Data Communications, and ACM Digital Library. *SIGCOMM’16: proceedings of the 2016 ACM Conference on Special Interest Groupon Data Communication : August 22-26, 2016, Florianopolis, Brazil.* English. OCLC: 987020496. 2016. URL: https://doi.org/10.1145/2934872 (visited on 03/01/2022).

[8] I.C. Wong and B.L. Evans. “Joint channel estimation and prediction for OFDM systems”. In: *GLOBECOM ’05. IEEE Global Telecommunications Conference, 2005.* St. Louis, MO, USA: IEEE, 2005, 5 pp.–2259. ISBN: 978-0-7803-9414-8. DOI: 10.1109/GLOCOM.

2005.1578065. URL: http://ieeexplore.ieee.org/document/1578065/ (visited on 02/28/2022).

[9] Wei Peng, Meng Zou, and Tao Jiang. "Channel Prediction in Time-Varying Massive MIMO Environments". In: *IEEE Access* 5 (2017), pp. 23938–23946. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2766091. URL: http://ieeexplore.ieee.org/document/8093592/ (visited on 03/01/2022).

[10] Wei Liu, Lie-Liang Yang, and L. Hanzo. "Recurrent Neural Network Based Narrowband Channel Prediction". In: *2006 IEEE 63rd Vehicular Technology Conference*. Vol. 5. Melbourne, Australia: IEEE, 2006, pp. 2173–2177. DOI: 10.1109/VETECS.2006.1683241. URL: http://ieeexplore.ieee.org/document/1683241/ (visited on 02/28/2022).

[11] Wei Jiang and Hans D. Schotten. "Neural Network-Based Fading Channel Prediction: A Comprehensive Overview". In: *IEEE Access* 7 (2019), pp. 118112–118124. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2937588. URL: https://ieeexplore.ieee.org/document/8813020/ (visited on 02/28/2022).

[12] Hwanjin Kim, Sucheol Kim, Hyeongtaek Lee, Chulhee Jang, Yongyun Choi, and Junil Choi. "Massive MIMO Channel Prediction: Kalman Filtering vs. Machine Learning". In: *arXiv:2009.09967 [cs, math]* (Sept. 2020). arXiv: 2009.09967. URL: http://arxiv.org/abs/2009.09967 (visited on 03/01/2022).

[13] Joel Poncha Lemayian and Jehad M. Hamamreh. "Massive MIMO Channel Prediction Using Recurrent Neural Networks". en. In: *RS Open Journal on Innovative Communication Technologies* 1.1 (Aug. 2020). DOI: 10.46470/03d8ffbd.80623473. URL: https://rs-ojict.pubpub.org/pub/massive-mimo-channel-prediction-using-recurrent-neural-networks (visited on 02/28/2022).

[14] Jie Wang, Ying Ding, Shujie Bian, Yang Peng, Miao Liu, and Guan Gui. "UL-CSI Data Driven Deep Learning for Predicting DL-CSI in Cellular FDD Systems". In: *IEEE Access* 7 (2019), pp. 96105–96112. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2929091. URL: https://ieeexplore.ieee.org/document/8764345/ (visited on 02/28/2022).

[15] Jide Yuan, Hien Quoc Ngo, and Michail Matthaiou. "Machine Learning-Based Channel Prediction in Massive MIMO With Channel Aging". In: *IEEE Trans. Wireless Commun.* 19.5 (May 2020), pp. 2960–2973. ISSN: 1536-1276, 1558-2248. DOI: 10.1109/TWC.2020.2969627. URL: https://ieeexplore.ieee.org/document/8979256/ (visited on 03/01/2022).

[16] Tianben Ding and Akira Hirose. "Fading Channel Prediction Based on Combination of Complex-Valued Neural Networks and Chirp Z-Transform". In: *IEEE Trans. Neural Netw. Learning Syst.* 25.9 (Sept. 2014), pp. 1686–1695. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2014.2306420. URL: http://ieeexplore.ieee.org/document/6755477/ (visited on 02/28/2022).

[17] Yuwen Yang, Feifei Gao, Geoffrey Ye Li, and Mengnan Jian. "Deep Learning based Downlink Channel Prediction for FDD Massive MIMO System". In: *arXiv:1908.03360 [cs, eess, math]* (Aug. 2019). arXiv: 1908.03360. URL: http://arxiv.org/abs/1908.03360 (visited on 02/28/2022).

[18] Arjun Bakshi, Yifan Mao, Kannan Srinivasan, and Srinivasan Parthasarathy. "Fast and Efficient Cross Band Channel Prediction Using Machine Learning". en. In: *The 25th Annual International Conference on Mobile Computing and Networking*. Los Cabos Mexico: ACM, Oct. 2019, pp. 1–16. ISBN: 978-1-4503-6169-9. DOI: 10.1145/3300061.3345438. URL: https://dl.acm.org/doi/10.1145/3300061.3345438 (visited on 03/01/2022).

[19] Markus Frohle, Themistoklis Charalambous, Ido Nevat, and Henk Wymeersch. “Channel Prediction With Location Uncertainty for Ad Hoc Networks”. In: *IEEE Trans. on Signal and Inf. Process. over Networks* 4.2 (June 2018), pp. 349–361. ISSN: 2373-776X, 2373-7778. DOI: 10.1109/TSIPN.2017.2705425. URL: https://ieeexplore.ieee.org/document/7931594/ (visited on 03/01/2022).

[20] Markus Frohle, L. Srikar Muppirisetty, and Henk Wymeersch. “Channel gain prediction for multi-agent networks in the presence of location uncertainty”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai: IEEE, Mar. 2016, pp. 3911–3915. ISBN: 978-1-4799-9988-0. DOI: 10.1109/ICASSP.2016.7472410. URL: http://ieeexplore.ieee.org/document/7472410/ (visited on 03/01/2022).

[21] J.M. Porta, J.J. Verbeek, and B.J.A. Kröse. “Active Appearance-Based Robot Localization Using Stereo Vision”. en. In: *Autonomous Robots* 18.1 (Jan. 2005), pp. 59–80. ISSN: 0929-5593. DOI: 10.1023/B:AURO.0000047287.00119.b6. URL: http://link.springer.com/10.1023/B:AURO.0000047287.00119.b6 (visited on 03/22/2022).

[22] Gift Nyamundanda, Lorraine Brennan, and Isobel Claire Gormley. “Probabilistic principal component analysis for metabolomic data”. en. In: *BMC Bioinformatics* 11.1 (Dec. 2010), p. 571. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-571. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-571 (visited on 03/22/2022).

[23] S. Oba, M.-a. Sato, I. Takemasa, M. Monden, K.-i. Matsubara, and S. Ishii. “A Bayesian missing value estimation method for gene expression profile data”. en. In: *Bioinformatics* 19.16 (Nov. 2003), pp. 2088–2096. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/btg287. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btg287 (visited on 03/22/2022).

[24] Erik Jenelius and Haris N. Koutsopoulos. “Urban Network Travel Time Prediction Based on a Probabilistic Principal Component Analysis Model of Probe Data”. In: *IEEE Trans. Intell. Transport. Syst.* 19.2 (Feb. 2018), pp. 436–445. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2017.2703652. URL: http://ieeexplore.ieee.org/document/7940066/ (visited on 03/22/2022).

[25] Zou Linfu, Pan Zhiwen, Jiang Huilin, Liu Nan, and You Xiaohu. “Deep Learning Based Downlink Channel Covariance Estimation for FDD Massive MIMO Systems”. In: *IEEE Communications Letters* 25.7 (2021), pp. 2275–2279. DOI: 10.1109/LCOMM.2021.3075725.

[26] Mostafa Hussien, Kim Khoa Nguyen, and Mohamed Cheriet. “PRVNet: Variational Autoencoders for Massive MIMO CSI Feedback”. In: *arXiv:2011.04178 [cs]* (Nov. 2020). arXiv: 2011.04178. URL: http://arxiv.org/abs/2011.04178 (visited on 04/24/2022).

[27] Zikun Liu, Gagandeep Singh, Chenren Xu, and Deepak Vasisht. “FIRE: enabling reciprocity for FDD MIMO systems”. en. In: *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. New Orleans Louisiana: ACM, Oct. 2021, pp. 628–641. ISBN: 978-1-4503-8342-4. DOI: 10.1145/3447993.3483275. URL: https://dl.acm.org/doi/10.1145/3447993.3483275 (visited on 05/03/2022).

[28] Wolfram Stacklies, Henning Redestig, Matthias Scholz, Dirk Walther, and Joachim Selbig. “pcaMethods – a Bioconductor package providing PCA methods for incomplete data”. In: *Bioinformatics* 23 (2007), pp. 1164–1167.

[29] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: (2013). Publisher: arXiv Version Number: 10. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114 (visited on 04/05/2022).

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. en. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836, 1476-4687. DOI: `10.1038/nature14539`. URL: `http://www.nature.com/articles/nature14539` (visited on 04/05/2022).

[31] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. eng. Softcover reprint of the original 1st edition 2006 (corrected at 8th printing 2009). Information Science and Statistics. New York, NY: Springer New York, 2016. ISBN: 978-1-4939-3843-8.

[32] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980. URL: `http://arxiv.org/abs/1412.6980` (visited on 04/06/2022).

[33] Hongquan Guo, Hoang Nguyen, Diep-Anh Vu, and Xuan-Nam Bui. “Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach”. en. In: *Resources Policy* 74 (Dec. 2021), p. 101474. ISSN: 03014207. DOI: `10.1016/j.resourpol.2019.101474`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0301420718306901` (visited on 04/06/2022).

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 978-0-262-03561-3.

[35] *By EugenioTL - Own work*. URL: `https://commons.wikimedia.org/w/index.php?curid=107231101`.

[36] Carl Doersch. “Tutorial on Variational Autoencoders”. In: *arXiv:1606.05908 [cs, stat]* (Jan. 2021). arXiv: 1606.05908. URL: `http://arxiv.org/abs/1606.05908` (visited on 04/11/2022).

[37] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518 (Apr. 2017). arXiv: 1601.00670, pp. 859–877. ISSN: 0162-1459, 1537-274X. DOI: `10.1080/01621459.2017.1285773`. URL: `http://arxiv.org/abs/1601.00670` (visited on 04/15/2022).

[38] Solomon Kullback. *Information theory and statistics*. eng. Reprint. Gloucester, Mass: Smith, 1978. ISBN: 978-0-8446-5625-0.

[39] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2017. URL: `https://www.R-project.org/`.

[40] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: `https://www.stats.ox.ac.uk/pub/MASS4/`.

[41] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[42] Francois Chollet et al. *Keras*. 2015. URL: `https://github.com/fchollet/keras`.

[43] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[44] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[45] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[46] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[47] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[48] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[49] Aravindh Krishnamoorthy and Deepak Menon. "Matrix Inversion Using Cholesky Decomposition". In: *arXiv:1111.4144 [cs]* (Oct. 2013). arXiv: 1111.4144. URL: http://arxiv.org/abs/1111.4144 (visited on 04/26/2022).

[50] fchollet. *Variational AutoEncoder*. May 2020. URL: https://keras.io/examples/generative/vae/.

[51] Michael Hahsler, Kurt Hornik, and Christian Buchta. "Getting Things in Order: An Introduction to the R Package seriation". In: *Journal of Statistical Software* 25.3 (2008), pp. 1–34. DOI: 10.18637/jss.v025.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v025i03.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[53] . *REGULATION (EU) 2015/2120 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*. Nov. 2015. URL: http://data.europa.eu/eli/reg/2015/2120/oj.

# A Appendix - Architecture Parameter Design Tables

Model: Encoder

| Layer (type) | Output Shape | Param | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 1280) | 0 | [ ] |
| dense_1 | (None, 309 ) | 395,829 | ['input_1[0][0]'] |
| z_mean (Dense) | (None, 50 ) | 15,500 | ['dense_1 [0][0]'] |
| z_log_var (Dense) | (None, 50 ) | 15,500 | ['dense_1 [0][0]'] |
| sampling_1 (Sampling) | (None, 50 ) | 0 | ['z_mean[0][0]', 'z_log_var[0][0]'] |

Total params: 426,829
Trainable params: 426,829
Non-trainable params: 0

Table A.1: Model summary for the Encoder NN (UE1)

Model: Decoder

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 50) | 0 |
| dense_2 (Dense) | (None, 50) | 2,550 |
| dense_3 (Dense) | (None, 3,840) | 195,840 |

Total params: 198,390
Trainable params: 198,390
Non-trainable params: 0

Table A.2: Model summary for Decoder NN (UE1)

Model: Encoder

| Layer (type) | Output Shape | Param | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 1,280) | 0 | [ ] |
| dense_1 | (None, 350 ) | 448,350 | ['input_1[0][0]'] |
| z_mean (Dense) | (None, 100 ) | 35,100 | ['dense_1[0][0]'] |
| z_log_var (Dense) | (None, 100 ) | 35,100 | ['dense_1[0][0]'] |
| sampling_1 (Sampling) | (None, 100 ) | 0 | ['z_mean[0][0]', 'z_log_var[0][0]'] |

Total params: 518,550
Trainable params: 518,550
Non-trainable params: 0

Table A.3: Model summary for the Encoder NN (UE2)

Model: Decoder

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input (InputLayer) | (None, 100) | 0 |
| dense_2 (Dense) | (None, 300) | 30,300 |
| dense_3 (Dense) | (None, 3,840) | 1,155,840 |

Total params: 1,186,140
Trainable params: 1,186,140
Non-trainable params: 0

Table A.4: Model summary for Decoder NN (UE2)

Model: Encoder

| Layer (type) | Output Shape | Param | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 1,280) | 0 | [ ] |
| dense_1 | (None, 309 ) | 395,829 | ['input_1[0][0]'] |
| z_mean (Dense) | (None, 100 ) | 31,000 | ['dense_1[0][0]'] |
| z_log_var (Dense) | (None, 100 ) | 31,000 | ['dense_1[0][0]'] |
| sampling_1 (Sampling) | (None, 100 ) | 0 | ['z_mean[0][0]', 'z_log_var[0][0]'] |

Total params: 457,829
Trainable params: 457,829
Non-trainable params: 0

Table A.5: Model summary for the Encoder NN (JointDS)

Model: Decoder

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input (InputLayer) | (None, 100) | 0 |
| dense (Dense) | (None, 300) | 30,300 |
| dense (Dense) | (None, 3,840) | 1,155,840 |

Total params: 1,186,140
Trainable params: 1,186,140
Non-trainable params: 0

Table A.6: Model summary for Decoder NN (JointDS)

# B Appendix - PPCA Results Plots

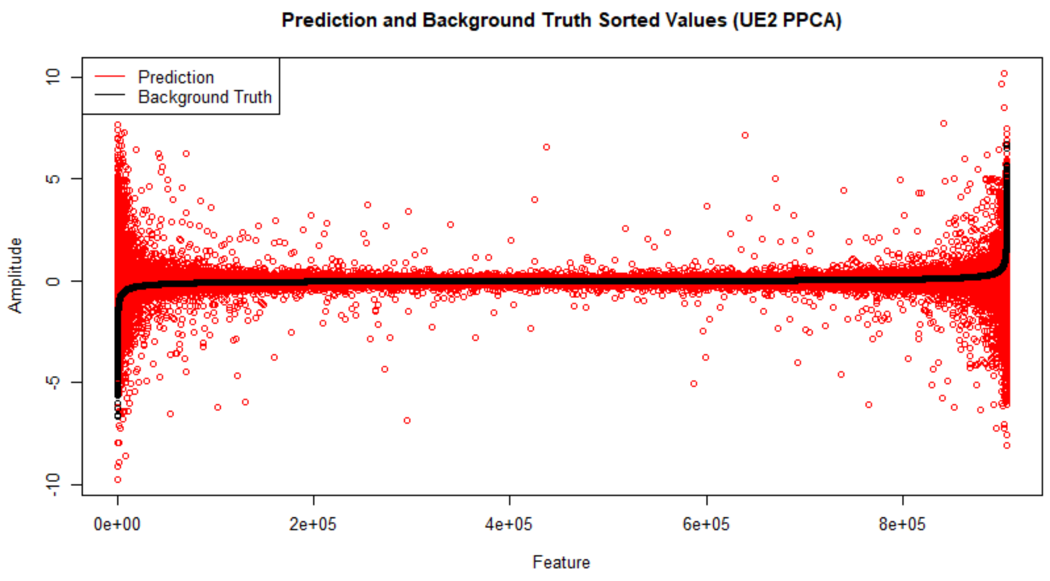


Figure B.1: Plot of the amplitudes for all predictions, PPCA - UE2 (Own elaboration) Plot of the amplitudes for all predictions. Note the black dots represent the actual observed Multipath Components (Re o Im part), whereas the red correspond to the predictions.

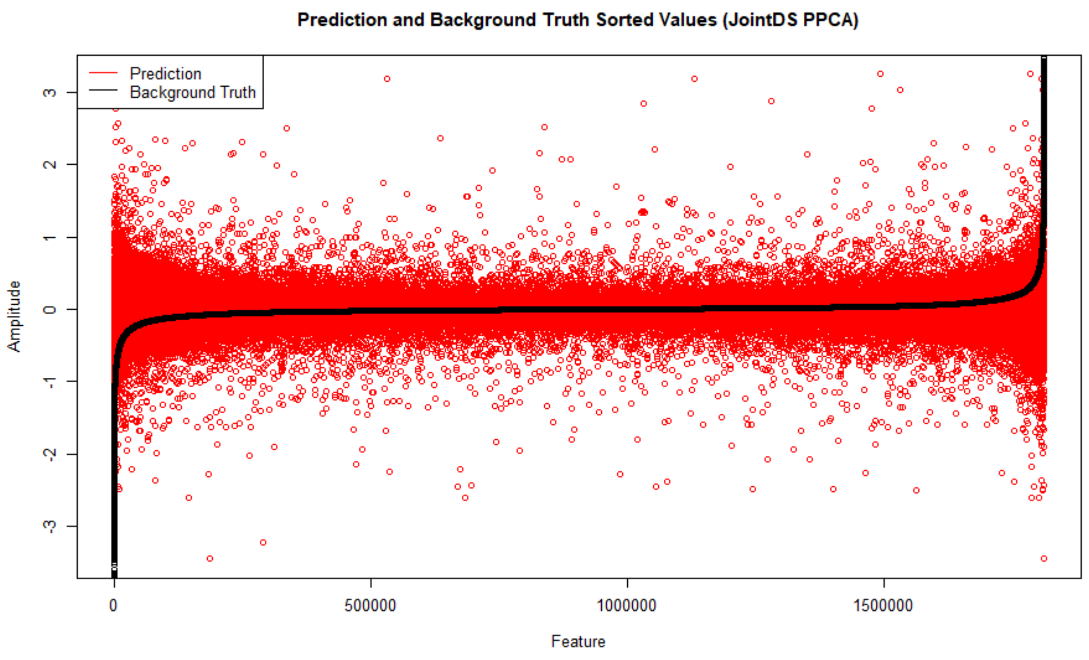


Figure B.2: Plot of the amplitudes for all predictions, PPCA - JointDS (Own elaboration)
Plot of the amplitudes for all predictions. Note the black dots represent the actual observed Multipath Components (Re o Im part), whereas the red correspond to the predictions.

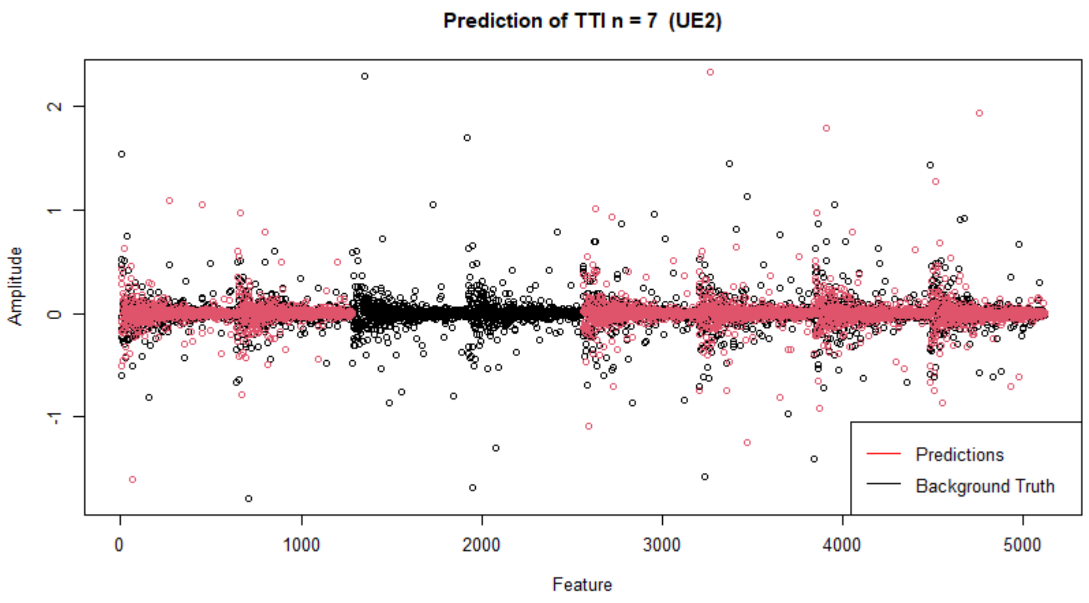


Figure B.3: Plot of the amplitudes for one TTI, PPCA - UE2 (Own elaboration)
Note the black dots represent the actual observed Multipath Components (Re or Im part), whereas the red correspond to the predictions. The segment of black values with no overlaid red is the observed Frequency Subband.

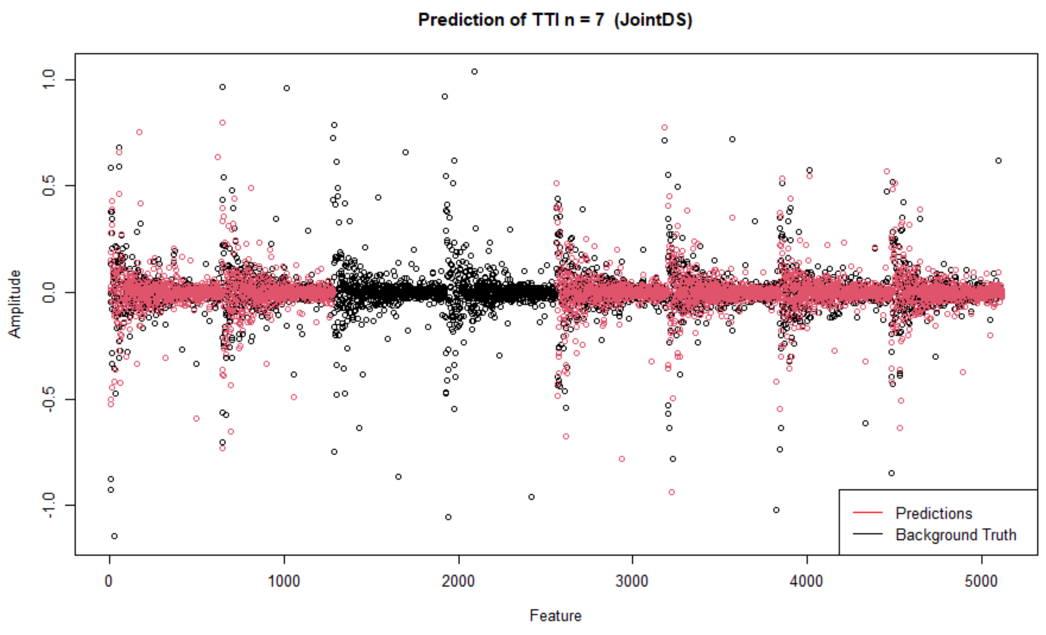


Figure B.4: Plot of the amplitudes for one TTI, PPCA - JointDS (Own elaboration) Note the black dots represent the actual observed Multipath Components (Re or Im part), whereas the red correspond to the predictions. The segment of black values with no overlaid red is the observed Frequency Subband.

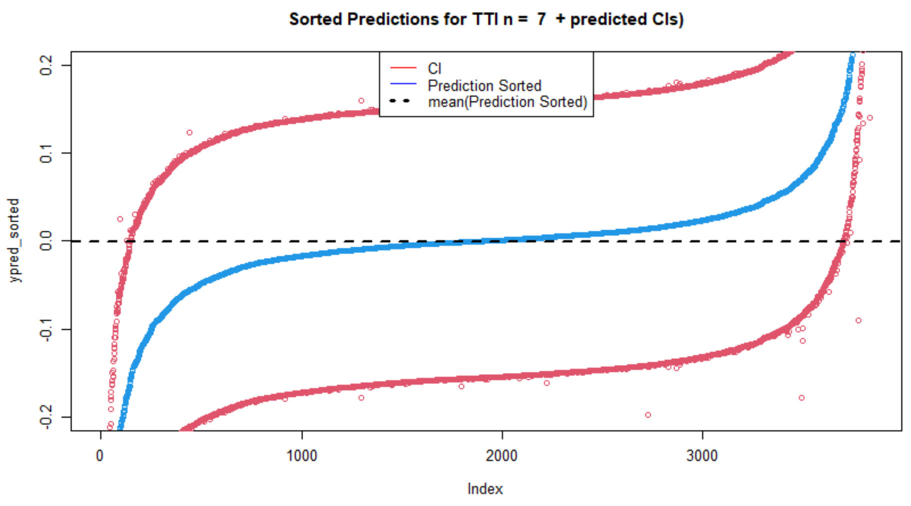


Figure B.5: This plot shows the sorted predicted values for 1 TTI along with the Confidence Intervals overlaid, UE2 (PPCA) (Own elaboration).

# C Appendix - VED Results Plots

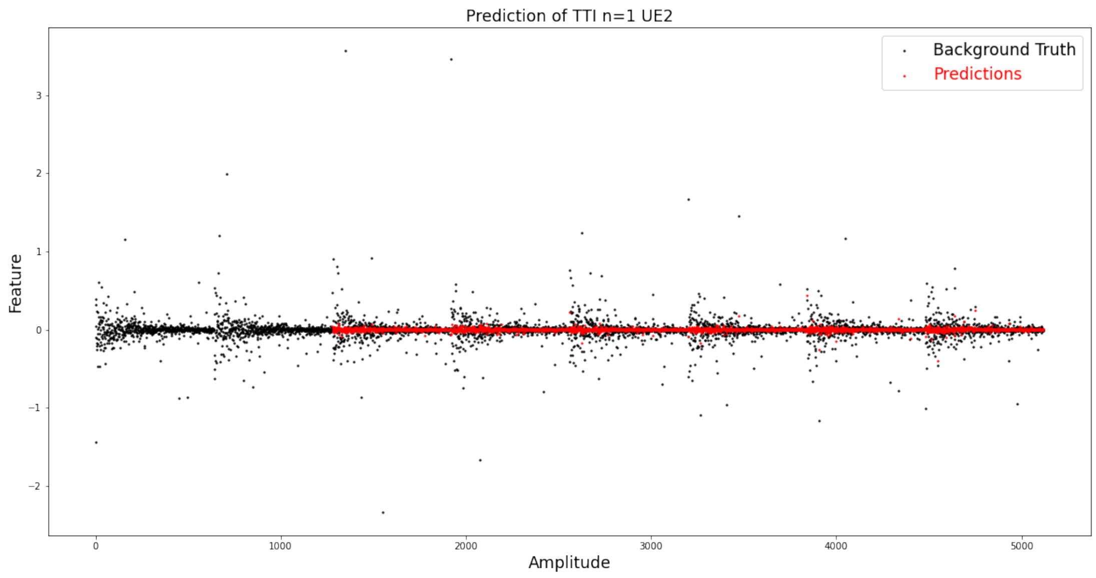


Figure C.1: Plot of the amplitudes for one TTI, VED - UE2 (Own elaboration)
Note the black dots represent the actual observed Multipath Components (Re or Im part), whereas the red correspond to the predictions. The segment of black values with no overlaid red is the observed Frequency Subband.

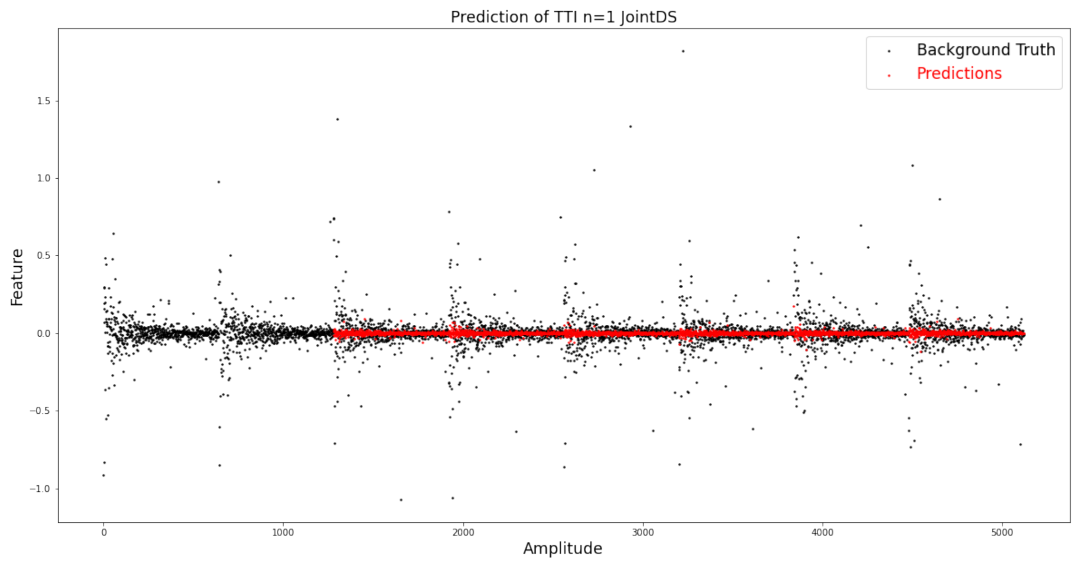


Figure C.2: Plot of the amplitudes for one TTI, VED - JointDS (Own elaboration)
Note the black dots represent the actual observed Multipath Components (Re or Im part), whereas the red correspond to the predictions. The segment of black values with no overlaid red is the observed Frequency Subband.

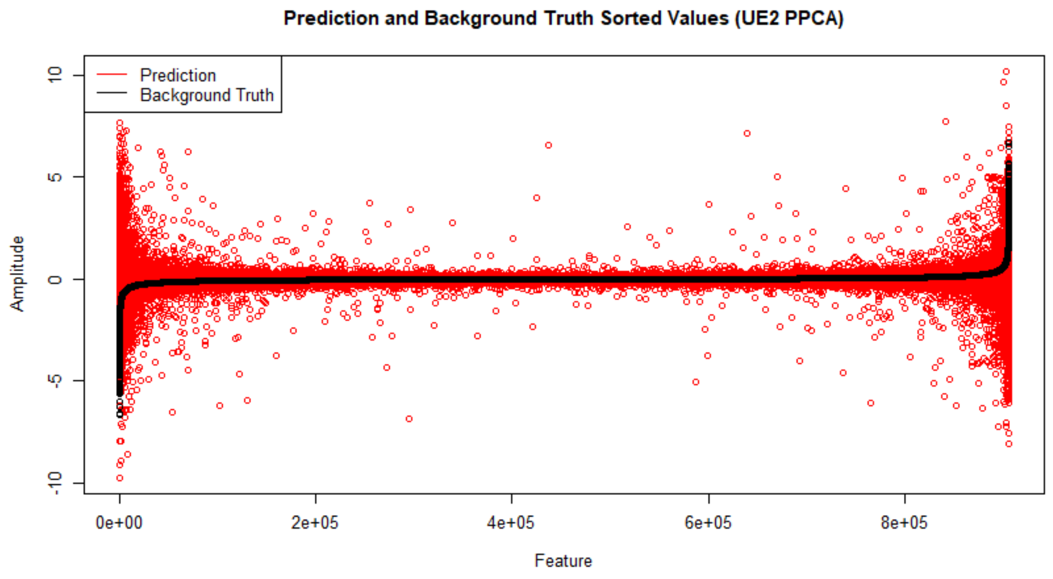


Figure C.3: Plot of the amplitudes for all predictions, VED - UE2 (Own elaboration)
Plot of the amplitudes for all predictions. Note the black dots represent the actual observed Multipath Components (Re o Im part), whereas the red correspond to the predictions.

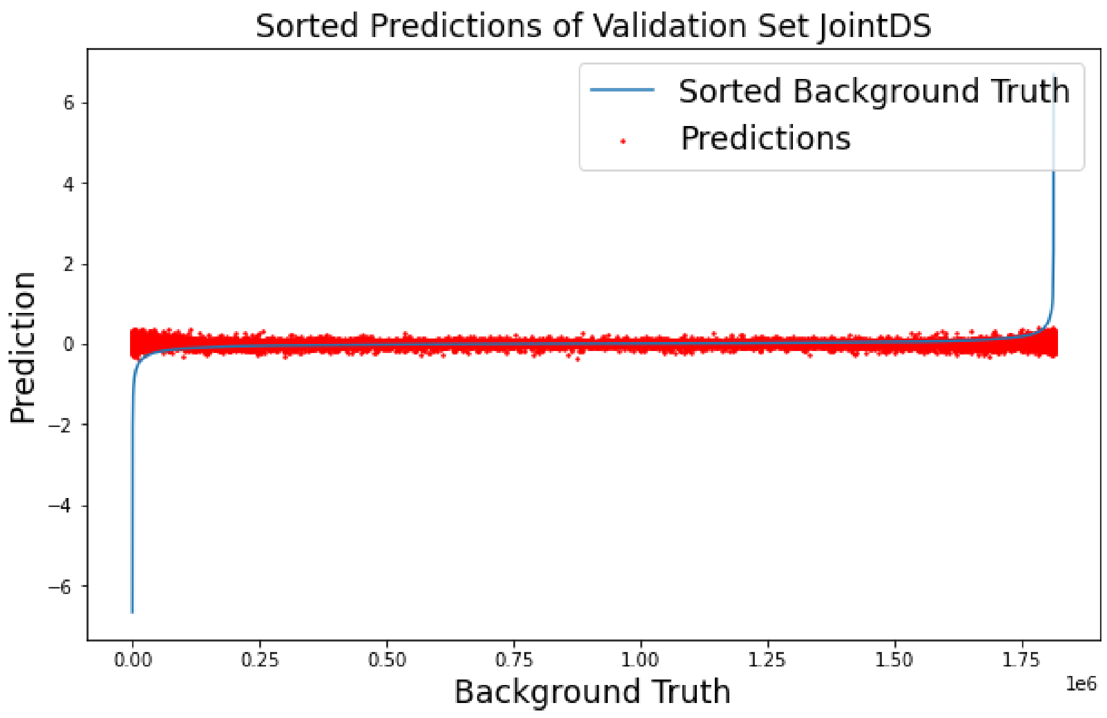


Figure C.4: Plot of the amplitudes for all predictions, VED - JointDS (Own elaboration) Plot of the amplitudes for all predictions. Note the black dots represent the actual observed Multipath Components (Re o Im part), whereas the red correspond to the predictions.