# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Uplink-Downlink Channel Prediction

Scaling the number of antennas up is a key characterstic of current and future wireless systems. Realizing the multiplexing and beamforming gains of large number of antennas requires channel knowledge. Usually channel feedback from users are used to get the channel knowledge. This results in signalling overhead and wastage of radio resources.

Suppose we know the channels between a user and a certain set of antennas at one frequency band, can we map this knowledge to the channels at a different set of antennas and at different frequency band? Essentially this mapping means that we can directly predict the downlink channel gains from the uplink channel gains, eliminating the downlink training and feedback overhead in co-located/distributed FDD massive MIMO systems. Following [2], we use deep neural networks (DNN) to approximate the mapping between uplink and downlink channel gains, and the freely available Deep MIMO Dataset [3] will be used for training and testing.

## 1.2 Performance Evaluation of AutoEncoder

Following [4], we present a communication system (transmitter, channel and receiver) as an autoencoder, and design an end to end reconstruction task that seeks to jointly optimize transmitter and receiver components in a single process. We compare the performance of end to end autoencoder for five different schemes. The key idea here is to represent transmitter, channel, and receiver as one deep neural network (NN) that can be trained as an autoencoder.

## 1.3 Organization of The Report

This chapter provides a background for the topics covered in this report. The rest of the chapters are organised as follows: Chapter 2 discuss the prior works. Chapter 3 discusses about the uplink downlink channel prediction in more detail. Chapter 4 discusses about AutoEncoder in more detail.

# Chapter 2

# Review of Prior Works

In this section we will discuss about the previous methods that were used.

## 2.1 Uplink-Downlink Channel Prediction

Estimating the channels at one frequency band using the channel knowledge at a different frequency band is attracting increasing interest [5], [6], [7], [8]. In [5], the parameters of the uplink channels, such as angle of arrival and path delay were estimated and used to construct the downlink channels at an adjacent frequency band. This frequency extrapolation concept was further studied in [6] where lower bounds on the mean squared error of the extrapolated channels were derived.[7], [8] proposed to leverage the channel knowledge at one frequency band to reduce the training overhead associated with design the mmWave beamforming vectors, leveraging the spatial correlation between the two frequency bands. A common feature of all the prior work [5], [6], [7], [8] is the requirement to first estimate some spatial knowledge (such as the angles of arrival) about the channels in one frequency band and then leverage this knowledge in the other frequency band. This channel parameter estimation process, however, is fundamentally limited by the system and hardware capability which highly affects the quality of the extrapolated channels.

# Chapter 3

# Uplink-Downlink Channel Prediction

In this chapter we will discuss more detail about the channel prediction. First we will start with the channel model.

## 3.1 Channel Model

Consider a system with one user $u$ at a distance $x_{u,\mathcal{A}_1}$ from the set of antennas $\mathcal{A}_1$ and at distance $x_{u,\mathcal{A}_2}$ from the set of antennas $\mathcal{A}_2$. User $u$ can communicate with $\mathcal{A}_1$ at frequency $f_1$ and antenna's $\mathcal{A}_2$ can communicate with $u$ at frequency $f_2$. We refer uplink when user sends the signal to antenna and downlink when antenna sends the signal to user. In Frequency division duplex(FDD) systems, uplink frequency will be $f_1$ and downlink frequency will be $f_2$.

Let $h_{f',u,a}$ denotes the channel from user $u$ to antenna $a$ when communicated with frequency $f'$. Then [9]

$$h_{f',u,a}(\tau) = \sum_{i=1}^{p} a_i e^{-j2\pi f' \tau_i} \delta(\tau - \tau_i)$$

where $a_i$ is the attenuation, $\tau_i = \frac{d_i}{c}$ is the propagation delay and $d_i$ denotes the distance travelled in path $i$, $p$ is the number of paths and $c$ is speed of light.

Converting this to frequency domain we will get

$$H_{f',u,a}(f) = \sum_{i=1}^{p} a_i e^{-j2\pi f' \tau_i} e^{-j2\pi f \tau_i}$$

We define channel vector in uplink as $H_{f_1,u,\mathcal{A}_1} = [H_{f_1,u,a_1}, ....., H_{f_1,u,a_{|\mathcal{A}_1|}}]^T$ which is a vector comprising $H$ from all the antennas in $\mathcal{A}_1$. Similarly we define channel vector at downlink as $H_{f_2,u,\mathcal{A}_2} = [H_{f_2,u,a_1}, ....., H_{f_2,u,a_{|\mathcal{A}_2|}}]^T$.

Formally the question we are trying to solve is can we map $H_{f_1,u,\mathcal{A}_1}$ to $H_{f_2,u,\mathcal{A}_2}$. Previous work [2] shows that this mapping exist under some assumptions and Deep neural nets (DNN) can be used to approximate this mapping from $H_{f_1,u,\mathcal{A}_1}$ to $H_{f_2,u,\mathcal{A}_2}$.

## 3.2 Dataset Description

In the previous section, we have formally introduced the problem and in this section we will describe the dataset used.

For the experiments we use the indoor distributed massive MIMO scenario I1 that is offered by DeepMIMO Dataset [3].The I1 scenario is available for frequencies 2.4 GHz (uplink) and 2.5 GHz (downlink) and it comprises of 10m x 10m room with 64 antennas attached to the ceiling at a height of 2.5m from the floor. The users are spread across two different grids as shown in Fig 3.1. The dataset contains the channel gain between every user and antenna at uplink and downlink frequencies. It contains million users with position vector of every user $u$ with respect to each antenna. Dataset was generated using ray tracing simulations that

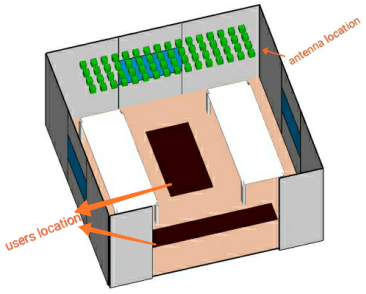


**Fig. 3.1**: 'I1' Dataset Scenario with users and antenna location marked

capture the geometry-based characteristics, such as the correlation between the channels

at different locations, and the dependence on the materials of the various elements of the environment, among others. The DeepMIMO dataset is generic (parametrized) and we can tune several parameters, such as the number of antennas, the array configuration, and the number of subcarriers. This allows the flexibility to study a wide range of network scenarios.

We have described the equation for continuous frequency domain above(in Section 3.1) but what dataset contains is $K$ equally spaced frequency domain samples (subcarriers in an OFDM system) with spacing $\triangle f$. So for user $u$ and antenna $a_i \in \mathcal{A}_1$ at uplink we will have a vector $[H_{f_1,u,a_i}(f_1+\triangle f), H_{f_1,u,a_i}(f_1+2\triangle f), ...., H_{f_1,u,a_i}(f_1+k\triangle f)]^T$. In case of downlink for user $u$ and antenna $a_i \in \mathcal{A}_2$ we will have a vector $[H_{f_2,u,a_i}(f_2+\triangle f), H_{f_2,u,a_i}(f_2+2\triangle f), ...., H_{f_2,u,a_i}(f_2+k\triangle f)]^T$. This is for one user antenna but when we consider for all the antennas then we have a matrix $H_{f_1,u,\mathcal{A}_1} = [H_{f_1,u,a_1}, ....., H_{f_1,u,a_{|\mathcal{A}_1|}}]^T$ where $H_{f_1,u,\mathcal{A}_1} \in \mathbb{C}^{|\mathcal{A}_1|\times K}$. In case of downlink we have a matrix $H_{f_2,u,\mathcal{A}_2} = [H_{f_2,u,a_1}, ....., H_{f_2,u,a_{|\mathcal{A}_2|}}]^T$ where $H_{f_2,u,\mathcal{A}_2} \in \mathbb{C}^{|\mathcal{A}_2|\times K}$.

Suppose we have $m$ users, each user will have $H_{f_1,u,\mathcal{A}_1} \in \mathbb{C}^{|\mathcal{A}_1|\times K}$ and $H_{f_2,u,\mathcal{A}_2} \in \mathbb{C}^{|\mathcal{A}_2|\times K}$ at uplink and downlink. We represent $H_{f_1,U,\mathcal{A}_1} \in \mathbb{C}^{m\times|\mathcal{A}_1|\times K}$, $H_{f_2,U,\mathcal{A}_2} \in \mathbb{C}^{m\times|\mathcal{A}_2|\times K}$ as the concatenation of user matrices at uplink and downlink respectively.

So our aim is **given** $H_{f_1,U,\mathcal{A}_1} \in \mathbb{C}^{m\times|\mathcal{A}_1|\times K}$**, can we estimate** $H_{f_2,U,\mathcal{A}_2} \in \mathbb{C}^{m\times|\mathcal{A}_1|\times K}$ **?**.

## 3.3 Neural Network Model

The model contains $L$ layers. The operation of each layer is given by

$$x_{l+1} = g_l(W_l x_l + b_l) = k_l(x_l)$$

where $W_l \in \mathbb{R}^{N_{l+1}\times N_l}, b_l \in \mathbb{R}^{N_{l+1}}$ are layer parameters and $g_l$ is called as activation function. For the input $x$ output after $L$ layers will be

$$o = k_{l-1}(.....k_2(k_1(x)))$$

The network parameters are $(W_l, b_l)_{l=1}^{L-1}$ which is to be trained. Usually it is done by constructing a loss function (which will depend on problem) and then minimizing the loss function will give the network parameters. The motivation for $g$ is to introduce non-linearity in the network. There are lot of possible choices for $g$ but the famous choice of $g$ is

$$g(x) = \max(0, x)$$

which is called as Rectified Linear Unit (ReLU).

Finding the minimum of loss function in closed form is not always possible, so optimization algorithms are used to find the minimum.

## 3.4 Preprocessing

Note that each entry in the matrices $H_{f_1,U,\mathcal{A}_1}$, $H_{f_2,U,\mathcal{A}_2}$ is a complex number but majority of deep learning softwares* perform real valued operations. To meet the requirement we concatenate real and imaginary parts of the matrix which will make $H_{f_1,U,\mathcal{A}_1} \in \mathbb{R}^{m\times|\mathcal{A}_1|\times 2K}$ and $H_{f_2,U,\mathcal{A}_2} \in \mathbb{R}^{m\times|\mathcal{A}_2|\times 2K}$. To make the neural network model perform efficiently, we have to preprocess its inputs(uplink channel gains) and outputs(downlink channel gains). We centrailize the uplink and downlink data with their mean and standard deviation. As we are estimating downlink data which means our problem is a regression problem we use normalized mean squared error (NMSE) to assess the quality of the output. The expression for NMSE is

$$l = \frac{1}{m}\sum_{i=1}^{m}\frac{\parallel y_{pred}^{(i)} - y_{true}^{(i)} \parallel_2}{\parallel y_{true}^{(i)} \parallel_2}$$

where $y_{pred}^{(i)} \in \mathbb{R}^n$ are predictions from the model , $y_{true}^{(i)} \in \mathbb{R}^n$ are true values for $i^{th}$ example, m is number of samples, $\parallel y \parallel_2$ denotes the $L_2$ norm.

*Examples include TensorFlow, Pytorch etc.

## 3.5 Benchmarking

Before actually fitting a neural net to the data we need a baseline to benchmark NMSE values, so that we can compare and comment about the performance of the neural network. A simple benchmark we tried is predicting downlink channel gains same as uplink channel gains. This is equivalent to the case when the neural net represents an identity function. The NMSE we got was 1.25, so if the NMSE with the neural network is close to this value then no learning is happening. Also, how much lower is the NMSE of neural network from this value is a measure of learning.

## 3.6 Simulation Details

The Following results are generated using this setup unless otherwise mentioned. We kept number of base stations to 1, antennas at base station as 1 and number of OFDM carriers as 64. The details of the simulation is given in the Table 3.1. For all the simulations we

| Parameter | Value |
|---|---|
| Scenario | I1_2p4, I1_2p5 |
| Number of BS's | 1 |
| Active Users | R1 to R502 |
| (num_ant_x, num_ant_y, num_ant_z) | (1,1,1) |
| number of OFDM carriers | 64 |

**Table 3.1**: I1 scenario details

consider $\mathcal{A}_1 = \mathcal{A}_2$ which implies $|\mathcal{A}_1| = |\mathcal{A}_2|$ unless otherwise mentioned. The total number of antennas at each set will be then equal to number of base stations multiplied by number of antennas at the base station. So given the scenarion in Table 3.1, $|\mathcal{A}_1| = |\mathcal{A}_2| = 1$ and $K = 64$, so effectively our channel gain matrix size for all users will be $H_{f_1,U,\mathcal{A}_1} \in \mathbb{R}^{m\times 1\times 128}$ and $H_{f_2,U,\mathcal{A}_2} \in \mathbb{R}^{m\times 1\times 128}$ which is equivalent saying that input size and output size of the neural network is 128.

## 3.7 Model Training

The total data is divided in the ratio $7:1:2$ representing training, validation and test data. For centralizing the inputs we have used training data mean and standard deviation. The network was trained for 100 epochs with mean squared error as loss function using ADAM [10] optimization algorithm and validation NMSE was recorded every epoch. Every time validation NMSE decreased the model was saved. After the completion, the best model was loaded back and then used to make predictions on test data. The learning rate for ADAM was 0.001 and weight decay was 0.0001.

## 3.8 Results

We trained a three layered neural network(256,512,512 as hidden layer size respectively) only with channel gains as input. From equation of channel model, we know that the attenuation $a_i$ is dependent on the user distance from the base station. So along with channel gains we have also added user, distance of user from base station and base station coordinates as the additional features. In I1 scenario the user's and base station $z$ coordinates stays constant, so we used only $x$ and $y$ coordinates. Table 3.2 shows the Test NMSE for different possible inputs.

| Experiment | Test NMSE |
|---|---|
| only channel gains | 0.1954 |
| channel gains + distance | 0.1379 |
| channel gains + distance + user coordinates | 0.0385 |
| channel gains + distance + user and bs coordinates | 0.0363 |
| channel gains + user and bs coordinates | 0.0347 |
| channel gains + distance + bs coordinates | 0.0766 |

**Table 3.2**: Test NMSE values for different inputs

### 3.8.1 Noise Robustness

We also checked the noise robustness of the neural network. We added noise to training , validation and test data. For channel gain matrix $H_{f_1,u,\mathcal{A}_1} \in \mathbb{C}^{|\mathcal{A}_1|\times K}$ of a user $u$ the noise variance is calculated as

$$\sigma_n^2 = \frac{1}{snr}\sum_{i=1}^{|\mathcal{A}_1|}\sum_{k=1}^{K}|H_{f_1,u,a_i}(f_1+k\Delta f)|^2$$

A complex Gaussian noise with above variance (implies that real and imaginary parts will have variance as $\sigma_n^2/2$) is sampled and added to the channel gain matrix. Point to note is that the noise variance will be different for different users. The results are shown in Table 3.3. Model is performing better at high SNR when compared with low SNR which is expected.

| SNR(dB) | 10 | 20 | 30 | no noise |
|---|---|---|---|---|
| Test NMSE | 0.1301 | 0.0834 | 0.0561 | 0.0385 |

**Table 3.3**: Test NMSE comparison for different noise levels

## 3.9 Antenna Mapping

Consider a DeepMIMO dataset parameters $\mathcal{S}$ that specifies (i) a number of BS antennas $M = M_x M_y M_z$ (ii) a set of OFDM subcarriers $\mathcal{K}$ at which the channels need to be calculated and (iii) and number of paths $L$. Then we have channel vector $h_k^{b,u} \in \mathbb{C}^M$ expressed as

$$h_k^{b,u} = \sum_{l=1}^{L}\sqrt{\frac{\rho l}{K}}e^{j(\vartheta_l^{b,u}+\frac{2\pi k}{K}\tau_l^{b,u}B)}a(\phi_{az}^{b,u},\phi_{el}^{b,u})$$

where $a(\phi_{az}^{b,u},\phi_{el}^{b,u})$ is array response vector of the base station. The array response vector is given by

$$a(\phi_{az}^{b,u},\phi_{el}^{b,u}) = a_z(\phi_{el}^{b,u})\otimes a_y(\phi_{az}^{b,u},\phi_{el}^{b,u})\otimes a_x(\phi_{az}^{b,u},\phi_{el}^{b,u})$$

In this setup, we made number of base station to 1 and number of OFDM carriers also to $1(K=1)$, but the antennas at base station to be 128 ($|\mathcal{A}_1| = |\mathcal{A}_2| = 128$), so effectively

| Experiment | Test NMSE |
|---|---|
| channel gains + distance + user coordinates | 0.0787 |
| channel gains + user and bs coordinates | 0.0759 |

**Table 3.4**: Test NMSE values for different inputs in antenna mapping setup

in this case channel gain matrix size for all users will $H_{f_1,U,\mathcal{A}_1} \in \mathbb{R}^{m\times 128\times 2}$ and $H_{f_2,U,\mathcal{A}_2} \in \mathbb{R}^{m\times 128\times 2}$ which equivalently says that input and output size of the neural network is 256. For this setup also we have tried adding features like distance, user and bs coordinates. The test NMSE for different possible inputs is shown in Table 3.4

# Chapter 4

# Performance Evaluation of AutoEncoder

Following [4], we present a communication system (transmitter, channel and receiver) as an autoencoder. Our goal is to assess the impact of using predicted downlink channel gains for transmit precoding on the error probability. We compare the performance of end to end autoencoder for five different schemes.

## 4.1 AutoEncoder Model

The aim of a communication system is to send the messages from transmitter to receiver through having the error rate to be minimum. In the autoencoder we input the message to encoder and then we update the weights of network such that output is close to the input. As we can see an autoencoder model can be represented as a communication system. In this model the transmitter is represented by the encoder and the receiver by the decoder. The channel effect is introduced by the channel layers.

Let us consider that we want to transmit $k$ bits of information. So there can be $M = 2^k$ possible symbols. Each symbol is one hot encoded as $M$-dimensional vector and represented as $s \in \mathbb{R}^M$. Transmitter needs $n$ channel uses for transmitting the symbol $s$. The

communication rate $R = k/n$ represents the number of bits we can transmit in one channel use. The input to the autoencoder is the symbol $s$ and channel gains $h_{in}$, which will be different for different schemes. From the Antenna Mapping Section 3.9, for each user $u$ we have channel gain matrix $H_{f_1,u,\mathcal{A}_1} \in C^{128\times 1}$ , $H_{f_2,u,\mathcal{A}_2} \in \mathbb{C}^{128\times 1}$ at uplink and downlink respectively as $|\mathcal{A}_1| = |\mathcal{A}_2| = 128$ and $K = 1$. Essentially in this case these are not matrices but vectors. Below is $h_{in}$ for five different schemes

1. $h_{in}$ = downlink channel gains : This scheme act's as a baseline and we expect it to have the best performance.

2. $h_{in}$ = predicted downlink channel gains: Given the low NMSE of channel prediction, we expect its performance to be close to the baseline.

3. $h_{in}$ = predicted downlink channel gains + plus additional features distance and user coordinates(only $x$ and $y$ coordinates): We expect this scheme to work better than Scheme 2 as additional features are seen to lower NMSE.

4. $h_{in}$ = uplink channel gains : In this scheme, the transmitter does joint channel prediction and encoding of input symbols.

5. $h_{in}$ = uplink channel gains + additional features distance and user coordinates(only $x$ and $y$ coordinates): We expect improvement over scheme 4 as additional features help to improve the estimation accuracy.

Predicted downlink channel gains here implies the prediction of NN when the input is uplink channel gains. All the channel gains mentioned in the schemes are taken from the simulation setup in Section 3.9. From that section we know that $|\mathbb{A}_1| = |\mathbb{A}_2| = |\mathcal{A}| = 128$. For the convenience we consider $h_{in} \in \mathbb{R}^p$ where $p = 2|\mathcal{A}|^*$ in schemes 1,3,4 and $p = 2|\mathcal{A}|+3^\dagger$ for schemes 2,5.

*concatenation of real and imaginary values will give $p = 2 \times 128 = 256$

†3 counts for additional features, effectively $p = 256 + 3 = 259$

### 4.1.1 Encoder

The input to the encoder(first block) will be vector $m = [s, h_{in}]^T \in \mathbb{R}^{M+p}$. The encoder realizes a transformation $f : \mathbb{R}^{M+p} \to \mathbb{R}^{2n|\mathcal{A}|}$ from $m$ to signal $x = f(m) \in \mathbb{R}^{2n|\mathcal{A}|}$. This is implemented as stack of multiple dense layers with last dense layer outputs required output size. After the transformation we impose the energy constraint which can be termed as normalization layer.

### 4.1.2 Channel

Channel layers are inserted between encoder and decoder represents the channel perturbation when the output of encoder $x$ passes the wireless channel as shown in Fig 4.1. The

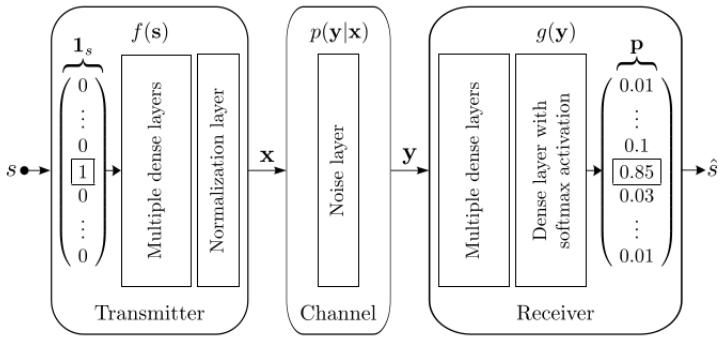


**Fig. 4.1**: AutoEncoder architecture taken from [1]. Difference here that input is $s + h_{in}$

channel we use here are the true downlink channel gains (simulation setup section 3.9). The channel layers transforms the input as

$$y_i = h^T x_i + n_i$$

where $h \in \mathbb{C}^{|\mathcal{A}|}$ represents the true downlink channel gains and $x_i \in \mathcal{C}^{|\mathcal{A}|}$ represents signal $x$ for $i^{th}$ use of channel, $n_i \in \mathbb{C}$ is the noise added by channel in $i^{th}$ use and $y_i \in \mathbb{C}$ is the output of channel for $i^{th}$ use. $x \in \mathcal{R}^{2n|\mathcal{A}|}$ can be transformed by converting it back to complex vector then $x \in \mathbb{C}^{n|\mathcal{A}|}$, now the complex vector $x$ can be thought as concatenated output of n channel uses. So $x = [x_1, x_2, .....x_i, .....x_n]^T$ where each $x_i \in \mathbb{C}^{|\mathcal{A}|}$. This is how $x_i$ is constructed from $x$. $n_i$ is complex gaussian noise with variance as

$$\sigma_n^2 = (2RE_b/N_0)^{-1}$$

where $E_b/N_0$ denotes the energy per bit ($E_b$) to noise power spectral density ($N_0$) ratio. Lastly we have $n$ $y_i$'s (output for each channel use) which can be concatenated to get the final output $y = [y_1, y_2, ....y_i, ...y_n]^T \in \mathbb{C}^n$ which then can be converted to real vector as $y \in \mathbb{R}^{2n}$.

### 4.1.3 Decoder

The receiver applies the transformation $g : \mathbb{R}^{2n} \to \mathbb{R}^M$ on noisy signal $y$ to get the estimate of message $\hat{s} \in \mathbb{R}^M$. The output of the decoder is a probability distribution consisting of $M$ probabilities. It can be implemented by stacking multiple dense layers and then on the final layer we can use softmax function. The network parameters are to be trained and the loss function we use is categorical cross entropy.

$$\mathcal{L}(s, \hat{s}) = -\sum_{j=1}^{M} s^{(j)} \log \hat{s}^{(j)}$$

where $s^{(j)}$ and $\hat{s}^{(j)}$ represents the $j^{th}$ component of original message and estimated message. Cross entropy for $m$ examples will be a simple average of individual cross entropies.

## 4.2 Preprocessing

Same as uplink-downlink prediction we centralize the input channel gains in all schemes. Also we standardize the channel gains even in the channel layer. For experiments we report block error rate and bit error rate. Block Error Rate (BLER) is a ratio of the number of erroneous blocks to the total number of blocks transmitted. The bit error ratio (BER) is the number of bit errors divided by the total number of transferred bits.

## 4.3 Model and Training Details

The autoencoder architecture is given Table 4.1

| Component | Layer | Activation Function | Input Dimension | Output Dimension |
|---|---|---|---|---|
| Transmitter | Dense | ReLU | $p + M$ | $2n\|\mathcal{A}\|$ |
| Transmitter | Dense | Linear | $2n\|\mathcal{A}\|$ | $2n\|\mathcal{A}\|$ |
| Transmitter | Normalization | | $2n\|\mathcal{A}\|$ | $2n\|\mathcal{A}\|$ |
| Channel | Fading | | $2n\|\mathcal{A}\|$ | $2n$ |
| Channel | Noise | | $2n$ | $2n$ |
| Receiver | Dense | ReLU | $2n$ | $M$ |
| Receiver | Dense | Softmax | $M$ | $M$ |

**Table 4.1**: AutoEncoder Architecture

The total data is divided in the ratio $7 : 1 : 2$ representing training, validation and test data. For centralizing the inputs we have used training data mean and standard deviation. For training we used $E_b/N_0 = 25$ dB and for testing we swept from $-20$ dB to 20 dB and calcuated BLER and BER. The network was trained till convergence using ADAM [10] algorithm. Every time validation categorical cross entropy decreased the model was saved. After the completion, the best model was loaded back and then used to make predictions on test data.

## 4.4 Results

We trained the network for all five schemes till convergence. The BLER curve for all schemes is shown in Fig 4.2 and BER curve for all schemes is shown in Fig 4.3. As expected scheme 1(dl-dl) performs best, then comes scheme 3(estdl+feats-dl) followed by scheme 2(estdl-dl), scheme 5(ul+feats-dl) and scheme 4(ul-dl).

Intuitively we can think scheme 4(ul-dl) and scheme 5(ul+feats-dl) as the joint prediction and encoding task which is more challenging. One more thing to observe that scheme 5(ul+feats-dl) is performing better than scheme 4(ul-dl) as additional features help in prediction task.

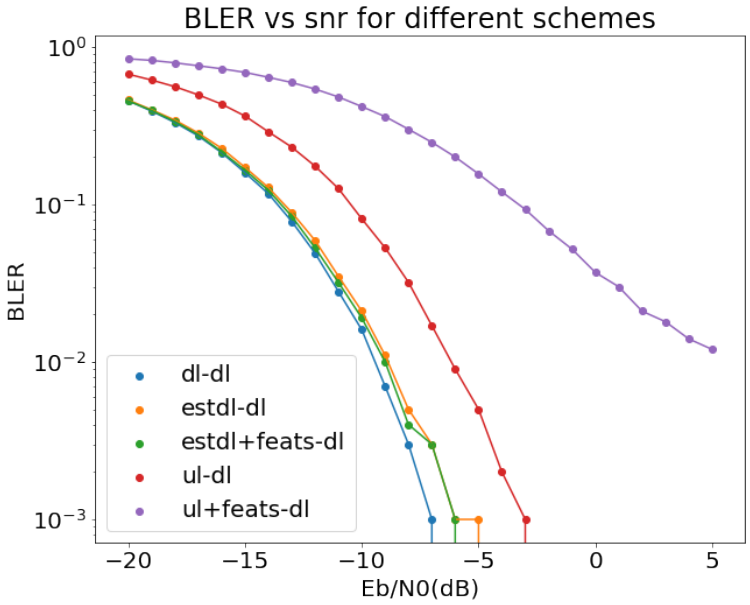


**Fig. 4.2**: BLER vs SNR for different schemes

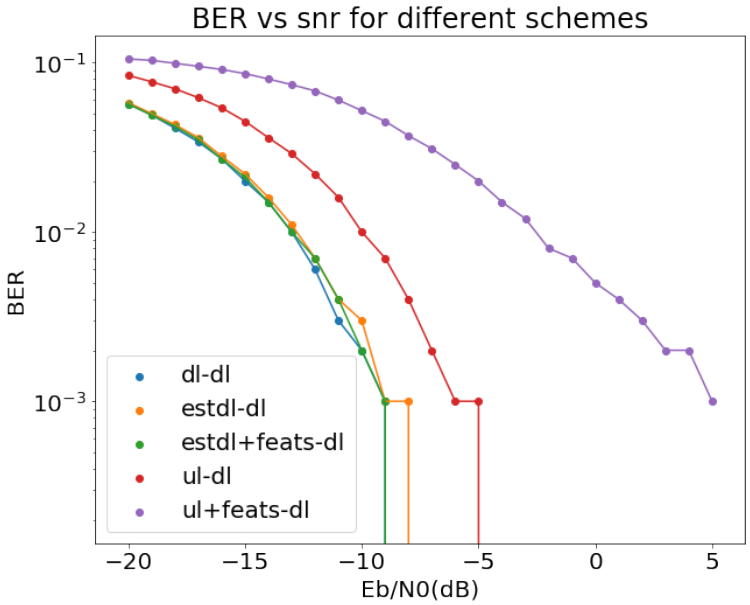


**Fig. 4.3**: BER vs SNR for different schemes

# Chapter 5

# Conclusion and Future Work

We have implemented NN for uplink - downlink channel mapping and improved the performance of NN by using additional features like distance and user coordinates. The feature added are quite intuitive as attenuation(channel model equation) depends on the distance. We also compared BLER and BER for five different channel prediction and symbol encoding and observed that the performance of joint channel prediction and symbol encoding improved with the proposed additional features. However, it was seen to be still quite away from the baseline.

Future work includes

1. Optimizing the encoder to get the performance of joint prediction and symbol encoding close to the baseline.

2. Extending the auto encoder architecture for multiple users. A basic idea is given at [1].

3. Evaluate the performance in the presence of channel estimation errors.

# References


[1] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[2] M. Alrabeiah and A. Alkhateeb, "Deep learning for tdd and fdd massive mimo: Mapping channels in space and frequency," *arXiv preprint arXiv:1905.03761*, 2019.

[3] A. Alkhateeb, "Deepmimo: A generic deep learning dataset for millimeter wave and massive mimo applications," *arXiv preprint arXiv:1902.06435*, 2019.

[4] J. Xu, W. Chen, B. Ai, R. He, Y. Li, J. Wang, T. Juhana, and A. Kurniawan, "Performance evaluation of autoencoder for coding and modulation in wireless communications," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.

[5] D. Vasisht, S. Kumar, H. Rahul, and D. Katabi, "Eliminating channel feedback in next-generation cellular networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 398–411.

[6] F. Rottenberg, R. Wang, J. Zhang, and A. F. Molisch, "Channel extrapolation in fdd massive mimo: Theoretical analysis and numerical validation," *arXiv preprint arXiv:1902.06844*, 2019.

[7] A. Ali, N. González-Prelcic, and R. W. Heath, "Millimeter wave beam-selection using out-of-band spatial information," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 1038–1052, 2017.

[8] F. Maschietti, D. Gesbert, and P. de Kerret, "Coordinated beam selection in millimeter wave multi-user mimo using out-of-band information," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[9] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.