Sejong University

Beamforming Design for Large-Scale Antenna Arrays Using Deep Learning

AUTHORS

LE XUAN HOANG - 24114545

2025-04-18

Contents

List of Figures

List of Tables

1 Abstract

Millimeter wave (mmWave) communication requires large antenna arrays and sophisticated beamforming to overcome high path loss, but practical hybrid architectures and imperfect channel state information (CSI) pose significant challenges. The Beamforming Neural Network (BFNN) proposed by Lin and Zhu [1] offers a promising deep learning (DL) approach to directly generate analog beamformers from estimated CSI, maximizing spectral efficiency (SE) while respecting hardware constraints. While effective, the sensitivity of the BFNN to hyperparameter choices and channel conditions requires systematic investigation. This project evaluates the performance sensitivity of the BFNN, reimplemented in PyTorch, focusing on its achievable SE. We systematically analyze the impact of key training parameters (activation functions: ReLU, Sigmoid, Tanh; learning rates: 0.001, 0.0001, 0.00001; training epochs: 100, 500, 1000) and channel characteristics (CSI quality via Pilot-to-Noise Ratio (PNR): -20, 0, 20 dB; estimated number of channel paths (Lest): 1, 2). Results indicate moderate sensitivity to the choice among common activation functions but strong dependence on learning rate and sufficient training epochs for convergence. Crucially, performance is significantly impacted by CSI quality (PNR), with higher PNR yielding substantially better SE. Furthermore, the accuracy of the channel path estimation (Lest) used by the CSI estimator feeding the BFNN significantly influences the final beamforming performance. These findings provide practical insights for configuring the BFNN and highlight the critical interplay between the DL model and the quality of the underlying channel estimation process.

For the full code implementation, please refer to the GitHub repository: `https://github.com/lexuanhoang120/Beam-forming-with-deep-learning`

2 Introduction

2.1 The Topic of the Project

Millimeter wave (mmWave) communication is essential for future high-data-rate wireless systems but suffers from high path loss, necessitating large antenna arrays and effective beamforming. Practical hardware limitations often lead to hybrid analog-digital architectures, where optimizing the phase-shifter-based analog beamformer under its constant modulus constraint presents a significant challenge. Furthermore, real-world systems must contend with imperfect Channel State Information (CSI), which can severely degrade the performance of traditional beamforming algorithms.

Deep Learning (DL) offers a promising data-driven alternative. Specifically, the Beamforming Neural Network (BFNN) proposed by Lin and Zhu [1] was designed to address these challenges. It directly generates the analog beamformer from estimated (imperfect) CSI, inherently satisfies hardware constraints, uses a performance-oriented loss function (maximizing Spectral Efficiency - SE), and demonstrated significant robustness and performance gains compared to traditional methods.

In this paper [1], the author proposed a deep learning (DL)-based beamforming (BF) design approach for large-scale antenna arrays. They developed a Beamforming Neural Network (BFNN) that can be trained to optimize the beamformer for maximizing spectral efficiency while considering hardware limitations and imperfect channel state information (CSI). The proposed BFNN demonstrates significant performance improvements and strong robustness to imperfect CSI compared to traditional BF algorithms. Key contributions of paper:

- A novel DL-based design approach that directly generates the optimized beamformer using estimated CSI as input.
- A newly designed loss function that aligns closely with spectral efficiency performance.
- A two-stage design approach to enhance BFNN robustness against imperfect CSI.

While the BFNN framework [1] shows considerable potential, its sensitivity to specific configuration choices and channel conditions requires further investigation, as the original work relied on empirically chosen parameters. This project aims to systematically analyze the performance of the BFNN, focusing on how its achievable Spectral Efficiency (SE) is impacted by variations in its architecture, training parameters, and key channel characteristics like CSI quality. The primary objectives are to:

- Investigate BFNN Architecture Sensitivity: Analyze the impact of network depth, width, and activation functions on SE.
- Analyze BFNN Training Parameter Impact: Evaluate the effect of learning rate and the number of training epochs on convergence and SE.

- Evaluate Robustness to Channel Conditions: Assess performance variations due to differing numbers of channel paths (L) and quantify robustness to CSI quality by varying the Pilot-to-Noise Ratio (PNR).

2.2 Objective of the Project

The primary goal is to analyze the impact of various parameters on the performance of the proposed Beamforming Neural Network (BFNN). The key performance metric will be spectral efficiency (SE), as used in the original paper [1].

- Validate the BFNN implementation by reproducing the baseline results.
- Provide comparative performance data and insights into parameter sensitivity and trade-offs.
- Gain a deeper understanding of the practical strengths and limitations of BFNN.
- Potentially offer guidelines for configuring the BFNN effectively in different scenarios.

After investigating these points, we will have some knowledge related to wireless communication.

2.3 Timeline

The project is conducted in a timeline of 6 weeks:

- Week 1: Paper selection and Literature review.
- Week 2: Writing proposal as Professor's assignment .
- Week 3: Implementation of the method.
- Week 4: Continue the implementation of the method.
- Week 5: Writing the report and implementation of related methods.
- Week 6: Finalization of the report and implementation of related methods.

3 Literature Review

Millimeter wave (mmWave) communication offers vast bandwidth for 5G and beyond but suffers from high path loss, requiring large antenna arrays and beamforming [1, 2, 3]. Due to the high cost and power consumption of fully digital beamforming, Hybrid Analog-Digital Beamforming (HBF) is a practical alternative [1, 2, 3, 4]. HBF uses fewer RF chains connected via an analog network (typically phase shifters), but optimizing this analog stage, especially under its constant modulus constraint, is challenging [2, 3, 4].

Traditional model-based optimization methods for the analog beamformer include codebook-based approaches like OMP [1], which are limited by codebook design, and continuous optimization techniques like manifold optimization [2, 3] or iterative algorithms [4], which can be computationally complex and may struggle with local optima. A major drawback shared by most traditional methods [1, 2, 3, 4] is their reliance on perfect Channel State Information (CSI), an unrealistic assumption leading to performance degradation in practice where only imperfect CSI is available [1, 5, 6].

Recognizing these limitations, researchers explored Deep Learning (DL) for beamforming [7, 8], leveraging its success in other wireless tasks [9, 10, 11, 12]. Early DL attempts for HBF [13, 14], however, often still assumed perfect CSI, struggled to enforce hardware constraints directly, or relied on codebooks.

A significant advancement was the Beamforming Neural Network (BFNN) proposed by Lin and Zhu [1]. This project builds directly on their work. The BFNN uniquely addresses prior limitations through several key contributions:

- It directly generates the analog beamformer using estimated (imperfect) CSI as input.
- It incorporates a custom "Lambda Layer" to inherently satisfy the constant modulus constraint.
- It uses a loss function based on maximizing Spectral Efficiency (SE), aligning training with the communication goal.
- It employs a specific training strategy designed to enhance robustness to imperfect CSI.

The BFNN demonstrated superior SE and robustness compared to traditional methods [1, 3, 4], especially under imperfect CSI. However, the original work [1] used empirically chosen network hyperparameters. A systematic analysis of how the BFNN's performance varies with its configuration and operating conditions is needed.

This project aims to fill that gap by conducting a comprehensive sensitivity analysis of the BFNN [1]. We will investigate how its spectral efficiency is affected by variations in network architecture (depth, width, activation functions), training parameters (learning rate, epochs), and channel conditions (number of paths, CSI quality via PNR), providing crucial insights for its practical optimization and deployment.

The detail of factors will be analyze:

- **Model parameters**

  - Activation Functions: Experiment with different activation functions in the dense layers (Sigmoid, Tanh, ReLU variants).
  - Learning Rate: Test different learning rates (0.001, 0.0001, 0.00001).
  - Number of Training Epochs: Test different epochs (100, 500, 1000).

- **Channel parameters**
  - Number of Estimated Channel Paths: Evaluate how different numbers of estimated channel paths impact performance (1, 2).
  - Pilot to Noise Ratio (PNR): Vary the PNR to simulate different levels of CSI quality (-20, 0, 20).

4 System Design

This section details the simulated communication system, channel model, beamforming architecture, and the specific configuration of the Beamforming Neural Network (BFNN) under investigation, along with the parameters used for performance evaluation.

4.1 System Model

We simulate a downlink millimeter wave (mmWave) communication scenario consisting of a single Base Station (BS) transmitting to a single User Equipment (UE), forming a Multiple-Input Single-Output (MISO) channel configuration.

- Base Station (BS): Equipped with a large-scale Uniform Linear Array (ULA) comprising Nt = 64 antennas with half-wavelength spacing. Critically, the BS operates with limited hardware complexity, featuring only a single Radio Frequency (RF) chain (NRF = 1).
- User Equipment (UE): Equipped with a single receive antenna.

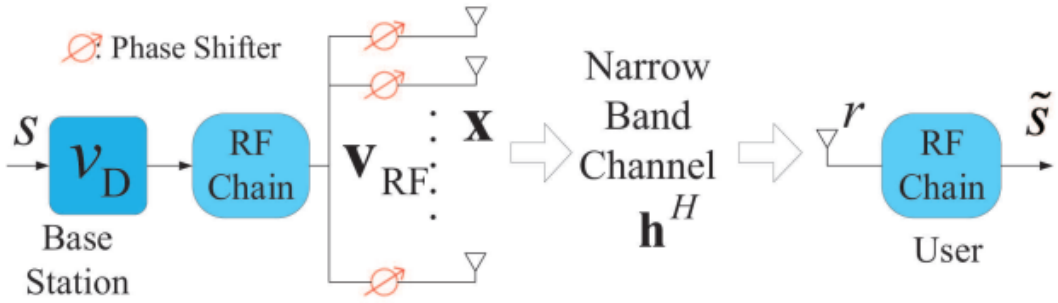

Figure 1: Diagram of an MISO mmWave system with on RF chain [1].

Consider the downlink of a narrowband multiple-input single-output (MISO) millimeter-wave (mmWave) system with an analog beamforming (BF) architecture, as illustrated in Fig. 1. A base station (BS) equipped with N_t antennas and a single RF chain transmits one data stream to a user terminal with a single receive antenna.

Let s denote the transmitted symbol with normalized average symbol energy, i.e., $E\{|s|^2\} = 1$. The symbol s is first multiplied by a scalar digital precoder v_D (since there is only one RF chain, v_D is a scalar), and then by an $N_t \times 1$ analog beamforming vector v_{RF}, which is implemented using phase shifters. The resulting transmitted signal is:

$$x = \mathbf{v}_{\text{RF}} v_D s \tag{1}$$

The received signal r at the user, passing through the MISO mmWave channel h, is given by:

$$r = h^H \mathbf{v}_{\text{RF}} v_D s + n \tag{2}$$

where n is additive white Gaussian noise (AWGN) modeled as a circularly symmetric complex Gaussian random variable with zero mean and variance σ^2, and h^H is the channel vector between the BS and the user.

We adopt the Saleh-Valenzuela channel model [6, 12], in which the channel vector h^H includes one line-of-sight (LoS) path and $(L-1)$ non-line-of-sight (NLoS) paths:

$$h^H = \sqrt{\frac{N_t}{L}} \sum_{l=1}^{L} \alpha_l a_t^H(\phi_l^t) \tag{3}$$

where α_l denotes the complex gain of the l-th path, and $a_t(\phi_l^t)$ represents the transmit antenna array response vector associated with the azimuth angle of departure ϕ_l^t. The term with $l = 1$ corresponds to the LoS component.

Although this work focuses on narrowband analog beamforming aided by deep learning (DL), the proposed design can be extended to broadband MIMO mmWave systems with hybrid beamforming (HBF).

We use spectral efficiency (SE) as the optimization objective, a widely used metric in beamforming literature [3, 4]. The SE for the studied system is given by:

$$R = \log_2 \left(1 + \frac{1}{\sigma^2} \left\| \mathbf{h}^H \mathbf{v}_{\text{RF}} v_D \right\|^2 \right). \tag{4}$$

Under the constant modulus constraint $|[\mathbf{v}_{\text{RF}}]_i|^2 = 1$ for $i = 1, \ldots, N_t$, and the transmit power constraint $\|v_{\text{RF}} v_D\|^2 \leq P$, it can be shown that the optimal value of v_D is $\sqrt{P/N_t}$. Substituting this value simplifies the optimization problem for $\mathbf{v}_{\text{RF}}$ as follows:

$$\begin{aligned} \text{maximize}_{v_{\text{RF}}} \quad & \log_2 \left(1 + \frac{\gamma}{N_t} \left\| \mathbf{h}^H \mathbf{v}_{\text{RF}} \right\|^2 \right) \\ \text{subject to} \quad & |[\mathbf{v}_{\text{RF}}]_i|^2 = 1, \quad i = 1, \ldots, N_t, \end{aligned} \tag{5}$$

where $\gamma = P/\sigma^2$ is the signal-to-noise ratio (SNR). Since SNR can typically be estimated more accurately than the channel state information (CSI), we assume the estimated SNR $\gamma_{\text{est}} = \gamma$, and focus our attention on handling imperfect CSI.

A crucial aspect of this simulation is the use of imperfect channel state information (CSI), which reflects practical deployment conditions.

- **Estimated CSI (h_{est})**: This $N_t \times 1$ vector represents the imperfect channel knowledge available at the base station (BS). It is obtained using a practical mmWave channel estimation method specifically, the hierarchical codebook-based estimator described in [5], as applied in [1]. The

quality of h_{est} depends on the Pilot-to-Noise Ratio (PNR) during the estimation phase. This estimated CSI serves as the primary input to the BFNN during both training and testing.

- **Perfect CSI (h)**: The true $N_t \times 1$ channel vector is assumed to be available only during the training phase of the BFNN. It is used exclusively to compute the loss function, enabling the network to learn a mapping from imperfect channel estimates to effective beamformers targeting the true channel.
- **Estimated SNR (γ_{est})**: It is assumed that the signal-to-noise ratio (SNR) is estimated with high accuracy, such that $\gamma_{\text{est}} = \gamma$.

4.2 BFNN Design

The core of the beamforming design is the BFNN proposed in [1]. This deep neural network learns to map the estimated CSI (h_{est}) and estimated SNR (γ_{est}) to the optimal analog beamforming vector ($\mathbf{v}_{\text{HF}}$).

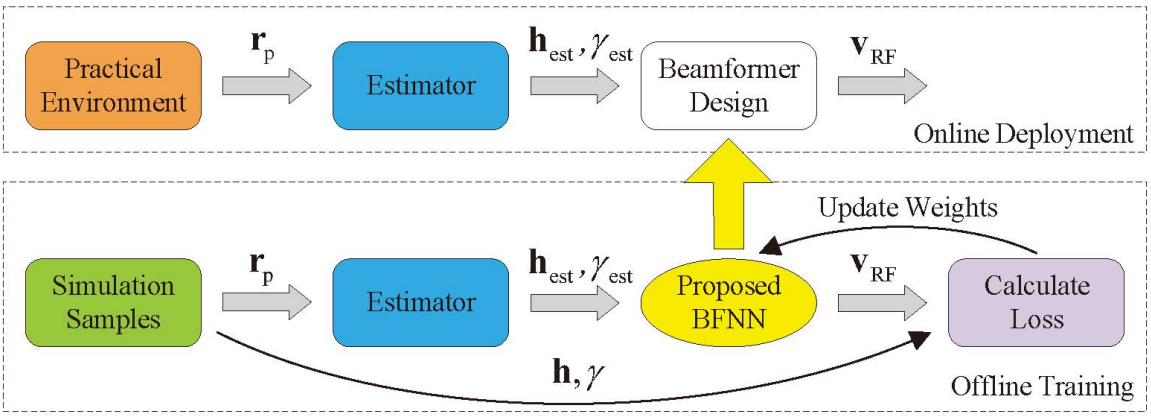

Figure 2: Illustration of the two-stage DL-based HBF design approach: offline training and online deployment [1].

The proposed beamforming design consists of two stages: offline training and online deployment, as depicted in Fig. 2.

- **Offline Training Stage**
  - Simulated channel samples, pilot symbols, and noise are generated based on the system model.
  - A practical mmWave channel estimator (from [5]) is used at the base station (BS) to obtain the estimated channel $\mathbf{h}_{\text{est}}$, using pilot transmissions and a hierarchical beamforming codebook.

  - The estimated channel $\mathbf{h}_{\text{est}}$ and estimated SNR γ_{est} (assumed to be equal to the true γ) are used as inputs to the BFNN.
  - The BFNN is trained to output the analog beamformer $\mathbf{v}_{\text{RF}}$ by minimizing a loss function based on spectral efficiency (SE), using the perfect CSI and true SNR only available during training.
  - This training enables the BFNN to learn how to approximate the ideal SE with perfect CSI and to be robust against CSI estimation errors.

- **Online Deployment Stage**
  - The same mmWave channel estimator is used to obtain $\mathbf{h}_{\text{est}}$ at the BS.
  - The trained BFNN takes $\mathbf{h}_{\text{est}}$ as input and directly outputs the optimized analog beamforming vector $\mathbf{v}_{\text{RF}}$.
  - No perfect CSI is needed in this stage; the BFNN operates using only imperfect CSI.

4.2.1 Input

The input to the BFNN is a real-valued vector of size $2N_t$, formed by concatenating the real and imaginary parts of the estimated channel vector $\mathbf{h}_{\text{est}}$. Specifically, when $N_t = 64$, the input dimension becomes $2 \times 64 = 128$.

4.2.2 Architecture

4.2.3 Loss function

The BFNN is trained offline using the Adam optimizer to minimize a loss function that is designed to maximize the average spectral efficiency (SE) across the training dataset. The loss function is defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} \log_2 \left(1 + \frac{\gamma_n}{N_t} \left\| \mathbf{h}_n^H \mathbf{v}_{\text{RF},n} \right\|^2 \right) \tag{6}$$

where:

- N is the total number of training samples,
- γ_n is the signal-to-noise ratio (SNR) of the n-th training sample,
- $\mathbf{h}_n$ is the channel vector of the n-th sample,
- $\mathbf{v}_{\text{RF},n}$ is the analog beamforming vector output by the BFNN for the n-th sample,
- N_t is the number of transmit antennas.

Note that minimizing this loss function directly corresponds to maximizing the average SE over the training dataset, making it well-aligned with the performance objective of the system.

4.2.4 Performance metric

The primary metric used to evaluate the performance of the proposed BFNN is the *Spectral Efficiency* (SE), which is measured in bits per second per Hertz (bits/s/Hz). For a given channel realization $\mathbf{h}$ and the corresponding analog beamformer $\mathbf{v}_{\mathrm{RF}}$, the instantaneous SE is calculated as:

$$\mathrm{SE} = \frac{1}{N} \sum_{n=1}^{N} \log_2 \left(\frac{\gamma_n}{N_t} \left\| \mathbf{h}_n^H \mathbf{v}_{\mathrm{RF},n} \right\|^2 \right) \tag{7}$$

where:

- N is the total number of training samples,
- γ_n is the signal-to-noise ratio (SNR) for the n-th sample,
- $\mathbf{h}_n^H$ is the Hermitian transpose of the channel vector for the n-th sample,
- $\mathbf{v}_{\mathrm{RF},n}$ is the analog beamforming vector generated by the BFNN for the n-th sample,
- N_t is the number of transmit antennas at the base station.

The final performance is typically evaluated by averaging the SE over multiple channel realizations and plotting it as a function of the SNR γ.

4.3 Assumption

System Model Validity: The project assumes the MISO downlink mmWave system model described in [1] is a representative and relevant scenario for studying analog beamforming. This includes:

- Single-user MISO configuration.
- Narrowband channel.
- Uniform Linear Array (ULA) at the BS (Nt=64).
- Single RF chain architecture (NRF=1), leading to pure analog beamforming.
- Phase shifter implementation, imposing the constant modulus constraint.

Channel Model Accuracy: The Saleh-Valenzuela channel model with the specific parameters used in [1] (Section IV: L=3 paths, specific power profiles, uniform AoD distribution) is assumed to adequately represent typical mmWave propagation characteristics for the purpose of this comparative analysis.

CSI Estimation Method: It's assumed that the method for obtaining imperfect CSI (hest) is consistent with that used in [1] (i.e., based on the hierarchical estimator from [5]), and that its quality is primarily determined by the Pilot-to-Noise Ratio (PNR). The simplification $\gamma_{\text{est}} = \gamma$ from [1] is also adopted.

Training Methodology: The two-stage training approach (offline training using hest as input and h for loss calculation) and the use of the SE-based loss function (Eq.6) with the Adam optimizer are assumed to be effective, as shown in [1].

Spectral Efficiency (SE) as Key Metric: SE is assumed to be the primary and sufficient performance metric for evaluating the beamformer's effectiveness, aligning with the objective function and the analysis in [1].

4.4 Simulation parameters

Table 1 summarizes the key parameters used in the simulation, including system, channel, CSI, BFNN architecture, and training parameters.

Table 1: Simulation parameters

Category	**Parameter**	**Description**	**Values**
System & Hardware	N_t	Number of BS antennas (ULA)	64
	N_{RF}	Number of RF chains at BS	1
	–	Number of UE antennas	1
	γ (SNR)	Signal-to-Noise Ratio (dB)	Varied (-20 to 20 dB)
Channel Model	L	True number of channel paths	3 (1 LoS, 2 NLoS)
	$\text{Var}(\alpha_l)$	Path gain variance	1 (LoS), $10^{-0.5}$ (NLoS)
CSI & Estimation	PNR	Pilot-to-Noise Ratio for $\hat{h}_{\text{est}}$ (dB)	-20, 0, 20
	L_{est}	Estimated number of paths (for context, see [1])	1, 2
	γ_{est}	Estimated SNR input to BFNN	$\gamma_{\text{est}} = \gamma$
BFNN Architecture	Input Dim	Size of BFNN input layer	128
	Depth	Number of Dense Hidden Layers	3
	Width	Neurons per Hidden Layer	$256 \to 128 \to 64$
	Activations	Activation functions (Hidden $\to$ Output Dense)	ReLU, Sigmoid, Tanh
	Output Dim	Output dimension before Lambda layer	64
BFNN Training	Optimizer	Algorithm for weight updates	Adam
	Learning Rate	Step size for optimizer	0.001, 0.0001, 0.00001
	Epochs	Number of training passes	100, 500, 1000
	N_{train}	Number of training samples	9×10^4
	N_{val}	Number of validation samples	10^4
	N_{test}	Number of testing samples	10^3
	Batch Size	Samples per weight update	256

5 Code Implementation

The implementation of this project is based on the work presented in [1], but has been rewritten using the PyTorch library[1]. All of the code was written in Python. The code for creating the dataset, as described in [1], was originally written in MATLAB. All the work from original paper is here[2].

5.1 Device specification

- **CPU:** Intel® Core™ i9-10900K CPU @ 3.70GHz×20
- **GPU:** 2×RTX 3090 (24GB)
- **Memory:** 128 GB
- **Storage:** 2.5 TB

5.2 Environmental Setup

These libraries must be installed prior to running the experiment. The version of Python used in this project is `3.10.16`. Below are some essential libraries required for the project:

- `torch==2.6.0`
- `pandas==2.2.3`
- `matplotlib==3.10.1`
- `seaborn==0.13.2`
- `scipy==1.15.2`
- `tensorboard`

We can install the required libraries using the following command in the terminal:

```
pip install -r requirements.txt
```

[1]https://pytorch.org/
[2]https://github.com/TianLin0509/BF-design-with-DL

5.3 Structure of project folder

Figure 3 describe the structure of project folder. Below is the description for each element in the project.

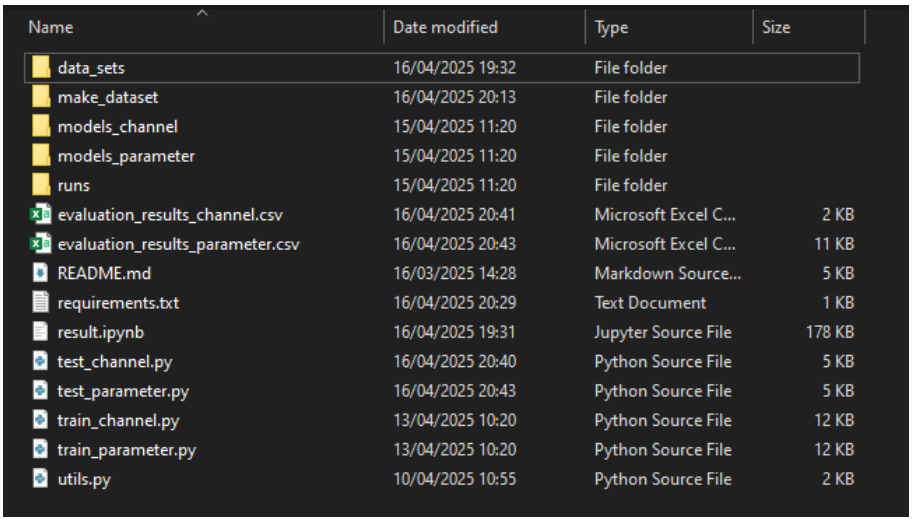

Figure 3: The structure of project folder

- **make_dataset/:** contains scripts or functions for creating datasets. It was writen in MATLAB. Please kindly refer to the **gen_samples.m** for details. The codes are based on the work [1, 5].
- **models_channel/:** stores trained model files results related to the channel scenario.
- **models_parameter/:** stores tranined model files related to the parameter scenario.
- **runs/:** store experiment logs and training metadata in Tensorboard.
- **train_set/:** includes training and testing data.
- **evaluation_results_channel.csv:** CSV file containing evaluation results for the model in channel scenarios.
- **evaluation_results_parameter.csv:** CSV file containing evaluation results for the model in parameter scenarios.
- **README.md:** providing an overview, setup instructions, and usage guide for the project.
- **requirements.txt:** lists all Python dependencies required for the project.
- **result.ipynb:** Jupyter Notebook, used for visualizing final results.

- **test_channel.py:** script for evaluating the trained models in channel scenarios.
- **test_parameter.py:** script for evaluating the trained models in parameter scenarios.
- **train_channel.py:** script for training the model in channel scenarios.
- **train_parameter.py:** script for training the model in parameter scenarios.
- **utils.py:** utility functions used across different scripts (data loading, computing rate, e.g.).

5.4 Replicating the Experiments

All files are organized to align with the experimental setup described in this report. To reproduce the results, simply follow the steps below in order. For a detailed understanding of each script's functionality, refer directly to the source code and in-line comments.

- **Step 1: Train the models for each scenario**
  Run the following commands in the terminal to train the models:
  - **Parameter-based scenario** see Section 6.1 for more details:

```
python train_parameter.py
```

  - **Channel-based scenario** see Section 6.2 for more details:

```
python train_channel.py
```

- **Step 2: Evaluate the trained models**
  After training, use the following commands to evaluate the models:
  - **Parameter-based scenario** see Section 6.1 for more details:

```
python test_parameter.py
```

  - **Channel-based scenario** see Section 6.2 for more details:

```
python test_channel.py
```

- **Step 3: Analyze the results**
  For result visualization and additional analysis, open and inspect the following Jupyter notebook:

```
result.ipynb
```

6 Performance Analysis

This section presents the simulation results evaluating the performance of the Beamforming Neural Network (BFNN). The analysis focuses on understanding the impact of key training parameters and channel characteristics on the network's ability to maximize spectral efficiency. For each experimental setup during the training process, the model achieving the best performance on a validation set was saved and subsequently evaluated on the testing dataset.

6.1 The Impact of Training Parameters

This subsection investigates the sensitivity of the BFNN's performance to fundamental training hyperparameters: activation function, learning rate, and the number of training epochs. The objective is to identify settings that lead to effective learning and high spectral efficiency.

6.1.1 Parameters Setup

For this analysis, the core BFNN architecture remained fixed, consistent with the configuration detailed in 4.2. The parameters systematically varied were:

- **Activation Function (Hidden Layers)**: ReLU, Sigmoid, Tanh.
- **Learning Rate**: 0.00001, 0.0001, 0.001.
- **Training Epochs**: 100, 500, 1000.

All other system and channel parameters were held constant at their baseline values (e.g., Nt=64, true L=3, specific PNR = 0).

6.1.2 Performance Metric: Summed Spectral Efficiency

Performance is primarily evaluated using the Summed Spectral Efficiency. This metric is calculated by averaging the Spectral Efficiency (SE) over the test dataset at each simulated SNR point (ranging from -20 dB to 20 dB) and then summing these average SE values across the entire SNR range. A higher sum indicates better overall performance. While useful for comparing aggregate trends, this metric averages potential performance variations at specific SNR regimes. Detailed results for all 27 combinations are provided in Table. 2.

Table 2: Experimental results with various Activations, Learning Rates, and Epochs

Experiment	Activation	Learning Rate	Epochs	Summed Spectral Efficiency
0	relu	0.00001	100	35.637077
1	relu	0.00001	500	37.192214
2	relu	0.00001	1000	37.248853
3	relu	0.0001	100	37.384750
4	relu	0.0001	500	37.651309
5	relu	0.0001	1000	37.434600
6	relu	0.001	100	37.986214
7	relu	0.001	500	37.826352
8	relu	0.001	1000	37.957547
9	sigmoid	0.00001	100	31.439638
10	sigmoid	0.00001	500	36.408837
11	sigmoid	0.00001	1000	37.435428
12	sigmoid	0.0001	100	37.500266
13	sigmoid	0.0001	500	38.421607
14	sigmoid	0.0001	1000	38.508183
15	sigmoid	0.001	100	38.935293
16	sigmoid	0.001	500	38.674803
17	sigmoid	0.001	1000	38.934986
18	tanh	0.00001	100	33.129587
19	tanh	0.00001	500	36.513843
20	tanh	0.00001	1000	37.390074
21	tanh	0.0001	100	37.720636
22	tanh	0.0001	500	38.136992
23	tanh	0.0001	1000	37.631534
24	tanh	0.001	100	38.088008
25	tanh	0.001	500	38.285456
26	tanh	0.001	1000	38.332740

6.1.3 Results and Discussion: Activation Functions

Table 3 aggregates the average Summed Spectral Efficiency achieved by each activation function, averaged across all tested learning rates and epoch counts. On average, all three activation functions (ReLU, Sigmoid, Tanh) yield remarkably similar overall performance. ReLU and Sigmoid are virtually tied, with Tanh performing only slightly lower on average. This suggests that, for this specific BFNN architecture and task, the choice of activation function (among these common options) might not be the most critical factor influencing the average outcome across various training settings.

Table 3: The summed spectral efficiency of different activation functions

Activation	Summed Spectral Efficiency
relu	37.368769
sigmoid	37.362116
tanh	37.247652

Table 4: The summed spectral efficiency on different learning rates and epochs

Epochs	Learning Rate	Summed Spectral Efficiency
100	0.00001	33.402101
100	0.0001	37.535217
100	0.001	38.336505
500	0.00001	36.704965
500	0.0001	38.069970
500	0.001	38.262203
1000	0.00001	37.358119
1000	0.0001	37.858106
1000	0.001	38.408424

6.1.4 Results and Discussion: Learning Rate and Epochs

Table 4 presents the average Summed Spectral Efficiency, illustrating the combined impact of learning rate and training duration. The results highlight the strong interplay between learning rate and training epochs, aligning with theoretical expectations:

- The lowest learning rate (0.00001) clearly leads to slow convergence, requiring substantially more epochs (500-1000) to approach the performance achieved much faster by higher rates.
- The highest tested learning rate (0.001) demonstrates rapid convergence, achieving the best average performance after only 100 epochs and maintaining high performance (reaching the overall peak average) up to 1000 epochs. This suggests 0.001 is close to an optimal rate for this setup, enabling efficient training.
- The intermediate learning rate (0.0001) offers a viable alternative, showing significant improvement between 100 and 500 epochs, after which performance tends to plateau, indicating convergence within that timeframe.
- Regarding epochs, 100 epochs is generally insufficient, especially with low learning rates. Performance significantly benefits from increasing training to 500 epochs. Further training to 1000

epochs yields diminishing returns for the faster learning rates (0.001, 0.0001), suggesting convergence is largely achieved around 500-1000 epochs.

6.1.5 Proposed Parameters for Subsequent Experience

Based on this analysis, aiming for efficient training and high performance, the following BFNN parameters were selected for the subsequent channel parameter investigations:

- **Activation Function**: ReLU (chosen as a robust, commonly used baseline that performed well on average).
- **Learning Rate**: 0.001 (demonstrated fastest convergence and highest overall performance).
- **Epochs**: 1000 (selected to ensure full convergence, capturing potentially the best performance achievable with the chosen learning rate).

6.2 The Impact of Channel Parameters

This subsection evaluates the BFNN's performance sensitivity to variations in the channel environment, specifically the quality of the Channel State Information (CSI) and the accuracy of channel sparsity estimation. The BFNN model used here employs the parameters determined in 6.1.5 (Activation=ReLU, LR=0.001, Epochs=1000).

6.2.1 Impact of CSI Quality PNR

Figure 4 plots the Spectral Efficiency (SE) versus SNR for three different Pilot-to-Noise Ratio (PNR) levels used during the channel estimation phase, which generates the hest input for the BFNN. Higher PNR corresponds to higher quality, more accurate CSI.

- The graph clearly shows that SE increases with SNR for all PNR levels, as theoretically expected. Crucially, it demonstrates a strong dependence on CSI quality: higher PNR consistently leads to higher SE across the entire SNR range. For instance, at an SNR of 20 dB, the SE achieved with PNR $= 20dB$ (11.15 bits/s/Hz) is substantially higher than with PNR $= 0dB$ (9.79 bits/s/Hz) and PNR $= -20dB$ (7.33 bits/s/Hz).
- This result confirms the importance of accurate channel estimation for effective beamforming. While the BFNN exhibits robustness, functioning even with poor CSI (PNR $= -20dB$), its performance ceiling is directly limited by the quality of the input hest. The performance gap widens at higher SNRs, indicating that accurate beam steering enabled by good CSI is increasingly critical when noise is less dominant. This underscores the practical need for efficient and accurate channel estimation mechanisms to fully leverage DL-based beamforming techniques like the BFNN.

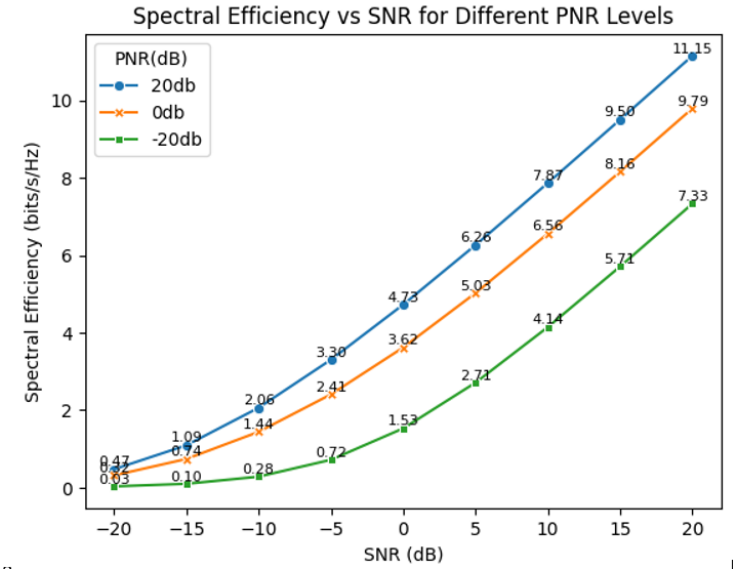

Figure 4: Spectral Efficiency vs. SNR for different PNR levels

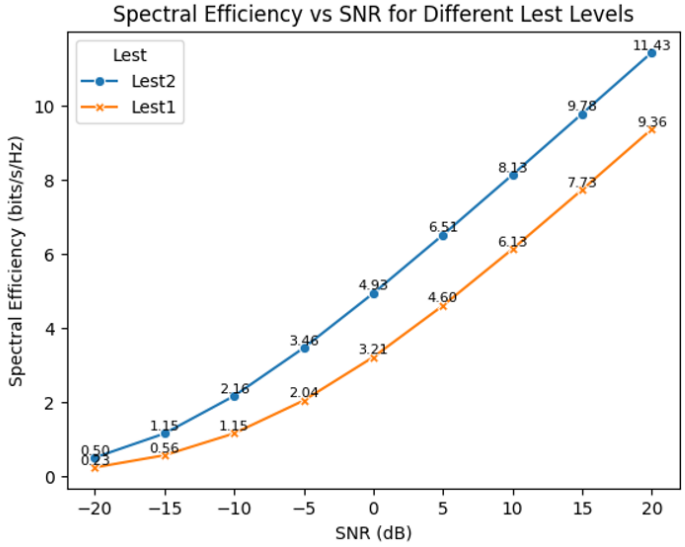

Figure 5: Spectral Efficiency vs. SNR for different L_{est} levels

6.2.2 Impact of Estimated Number of Paths L_{est}

Figure 5 investigates the effect of inaccuracies in estimating the channel sparsity (number of paths, L) during the channel estimation step. The graph compares performance when the estimator assumes $L_{\text{est}} = 1$ or $L_{\text{est}} = 2$ paths.

As with PNR, SE increases with SNR for both L_{est} values. The key finding is that assuming $L_{\text{est}} = 1$ consistently yields higher SE compared to assuming $L_{\text{est}} = 1$. The performance gap is significant, reaching over 2 bits/s/Hz at high SNRs.

This demonstrates that the performance of the subsequent BFNN beamforming is sensitive to the accuracy of the channel model parameters assumed by the channel estimator. When the estimator assumes $L_{\text{est}} = 1$, it likely captures only the dominant path, resulting in a less accurate hest compared to when it assumes $L_{\text{est}} = 2$, which can capture two paths. This more accurate hest (from $L_{\text{est}} = 2$) allows the BFNN, trained on the true channel characteristics, to generate a more effective beamformer. While the system shows robustness to this underestimation (performance with $L_{\text{est}} = 1$ is still reasonable), striving for accurate sparsity estimation during the CSI acquisition phase is crucial for maximizing the overall system performance enabled by the BFNN.

7 Conclusion

This project conducted a systematic performance analysis of the Deep Learning-based Beamforming Neural Network (BFNN) for analog beamforming in mmWave MISO systems, as proposed in [1]. By re-implementing the framework in PyTorch and simulating various scenarios, we investigated the sensitivity of the BFNN's spectral efficiency (SE) to its training hyperparameters and key channel characteristics.

Our analysis of training parameters revealed that while the choice between common activation functions (ReLU, Sigmoid, Tanh) had a relatively minor impact on the overall average performance, the learning rate and number of training epochs were critical. A higher learning rate (0.001) facilitated rapid convergence, achieving near-optimal performance within fewer epochs compared to lower rates. Sufficient training (500-1000 epochs for faster learning rates) was necessary to realize the BFNN's full potential. Based on these findings, ReLU activation, a learning rate of 0.001, and 1000 epochs were identified as effective parameters for this specific architecture and dataset.

The investigation into channel parameters underscored the BFNN's dependence on the quality of its input. Performance, measured by SE, degraded significantly as the quality of the estimated CSI decreased (lower PNR). This confirms that while the BFNN exhibits robustness to imperfect CSI compared to traditional methods, maximizing its benefits necessitates accurate channel estimation. Furthermore, the results demonstrated sensitivity to the accuracy of the channel model assumptions made during the channel estimation phase. Using an estimated number of paths (Lest=2) closer to the true channel sparsity yielded notably higher SE compared to a less accurate assumption (Lest=1), indicating that the performance relies not only on the BFNN itself but also on the fidelity of the upstream estimation process.

In summary, the BFNN stands as a powerful and robust data-driven approach for mmWave beamforming, capable of learning effective beamforming strategies directly from imperfect CSI. However, its practical deployment requires careful tuning of training hyperparameters for efficient learning and, crucially, relies on high-quality channel estimation that accurately reflects the underlying channel structure. Future work could explore architectural variations, more complex multi-user scenarios, or the integration of the BFNN with adaptive or more sophisticated CSI acquisition techniques.

References

[1] T. Lin and Y. Zhu, "Beamforming design for large-scale antenna arrays using deep learning," *IEEE Wireless Communications Letters*, vol. 9, no. 1, pp. 60–64, 2020.

[2] T. Lin, J. Cong, Y. Zhu, J. Zhang, and K. B. Letaief, "Hybrid beamforming for millimeter wave systems using the mmse criterion," *IEEE Transactions on Communications*, vol. 67, pp. 3693–3708, May 2019.

[3] X. Yu, J. Shen, J. Zhang, and K. B. Letaief, "Alternating minimization algorithms for hybrid precoding in millimeter wave mimo systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, pp. 485–500, Apr. 2016.

[4] F. Sohrabi and W. Yu, "Hybrid digital and analog beamforming design for large-scale antenna arrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, pp. 501–513, Apr. 2016.

[5] A. Alkhateeb, O. E. Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, pp. 831–846, Oct. 2014.

[6] X. Gao, L. Dai, S. Han, C.-L. I, and X. Wang, "Reliable beamspace channel estimation for millimeter-wave massive mimo systems with lens antenna array," *IEEE Transactions on Wireless Communications*, vol. 16, pp. 6010–6021, Sep. 2017.

[7] S. DÃ¶rner, S. Cammerer, J. Hoydis, and S. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, pp. 132–143, Feb. 2018.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[9] H. Ye, G. Y. Li, and B. Juang, "Power of deep learning for channel estimation and signal detection in ofdm systems," *IEEE Wireless Communications Letters*, vol. 7, pp. 114–117, Feb. 2018.

[10] X. Gao, S. Jin, C. Wen, and G. Y. Li, "Comnet: Combination of deep learning and expert knowledge in ofdm receivers," *IEEE Communications Letters*, vol. 22, pp. 2627–2630, Dec. 2018.

[11] C. Wen, W. Shih, and S. Jin, "Deep learning for massive mimo csi feedback," *IEEE Wireless Communications Letters*, vol. 7, pp. 748–751, Oct. 2018.

[12] H. He, C. Wen, S. Jin, and G. Y. Li, "Deep learning-based channel estimation for beamspace mmwave massive mimo systems," *IEEE Wireless Communications Letters*, vol. 7, pp. 852–855, Oct. 2018.

[13] H. Huang, Y. Song, J. Yang, G. Gui, and F. Adachi, "Deep-learning-based millimeter-wave massive mimo for hybrid precoding," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 3027–3032, Mar. 2019.

[14] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tujkovic, "Deep learning coordinated beamforming for highly-mobile millimeter wave systems," *IEEE Access*, vol. 6, pp. 37328–37348, 2018.