Horizon Europe Framework Programme

CENTRIC

Towards an AI-native, user-centric air interface for 6G networks

D3.4 CENTRIC's AI-Based MIMO Toolset

Contractual Delivery Date:	30-09-2024
Actual Delivery Date:	30-09-2024
Editor: *(name, organization)*	Ramoni Adeogun, Aalborg University
Deliverable nature:	OTHER
Dissemination level:	PU
Version:	1.0

Keywords: MIMO, AI, Beam management, Neural receiver, Transfer learning, CSI compression

ABSTRACT

This report presents a description of five open-source repositories developed as part of CENTRIC Work Package 3 (WP3)'s activities and published as CENTRIC's AI-based MIMO toolset in deliverable D3.4. The repositories include software implementation and documentation of simulation environments and AI-based MIMO algorithms developed in WP3 as part of the physical layer processing component of the AI native Air-Interface (AI-AI) concept which CENTRIC is developing. Each repository focuses on specific MIMO-AI algorithm such as reinforcement learning for beam management in Integrated Sensing and

This project has received funding from the European Union's Smart Networks and Services Joint Undertaking (SNS JU) under grant agreement No. 101096379.

Communication (ISAC) scenarios, AI techniques for wide to narrow beam prediction, multi-user MIMO neural network-based receiver, transfer learning techniques for neural receiver and learning based beam alignment. This report provides detailed description of the problem addressed, system model and simulation environment, the developed AI algorithm, and usage example and results obtained from each of the repositories. These repositories offer a comprehensive set of MIMO-AI solutions allowing reproducibility of CENTRIC's research output while serving as the basis for further development of AI-AI physical layer methods.

Disclaimer

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101096379. This publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Impressum

Full project title: Towards an AI-native, user-centric air interface for 6G networks

Short project title: CENTRIC

Number and title of the work package: WP3 AI-AI Physical Layer Methods

Number and title of task: T3.2: End-to-End Air-Interface Learning, T3.3: AI-based Communication and Sensing in mmWave Communications & T3.4: Nextgeneration AI-empowered MIMO Communications

Document title: CENTRIC's AI-Based MIMO Toolset

Editor: Ramoni Adeogun, Aalborg University

Work-package leader: Ramoni Adeogun, Aalborg University.

Executive summary

This report presents a comprehensive overview of five open-source software repositories developed as part of CENTRIC Work Package 3 (WP3) research on AI-AI physical layer methods. The repositories which are referred to as CENTRIC's MIMO-AI Toolset contains implementation and documentation of novel MIMO AI algorithms and simulation environments developed in CENTRIC. The publication of these open-source software represents a major milestone towards reproducibility of CENTRIC's research outputs and further development of AI based solutions for the physical layer of future wireless networks.

The first repository contains reinforcement learning based solutions for resource allocation in millimetre wave (mmWave) Integrated Sensing and Communication (ISAC) scenarios. The repository provides a simulation environment for beam management in a vehicular wireless network comprising of a base station (BS) serving multiple vehicles on appropriately allocated beam. Python implementation of a novel reinforcement learning algorithm based on proximal policy optimization for allocating resources to either communication (beam management) and sensing is also included in the repository allowing for evaluation and benchmarking of the included methods as well as easy integration of new and efficient AI algorithms for resource allocation in mmWave ISAC scenarios.

The second repository documents real time multi-user MIMO neural network (NN)-based receivers that comply with 5G New Radio (5G NR), positioning them as enabling technology for novel applications such as re-trainable site-specific base stations and pilotless communications. The current version of the receiver developed in WP3 is a flexible multi-user MIMO (MU-MIMO) receiver with 5G NR physical uplink shared channel (PUSCH) compatibility. The receiver architecture combines graph neural networks (GNN) and convolutional neural networks (CNN), allowing flexibility in handling varying user numbers and sub-carrier configurations without retraining. In this repository which is prepared and managed by NVIDIA, a comprehensive implementation and documentation including simulation examples and tutorial-like materials are made available publicly.

In the third repository, implementation of transfer learning techniques applied to the neural receiver are provided. The provided codes allow for evaluating the transferability of neural receivers trained in specific environments or for specific tasks to other environments or tasks and the potential of various transfer learning techniques to minimize the amount of data required for retraining of neural receivers for new environments or tasks. This repository will make it possible to easily integrate and evaluate new transfer learning techniques.

The fourth repository presents implementation of a novel learning-based algorithm for the joint, two-sided beam alignment in mmWave communication systems. The novel scheme implemented in this repository combines the benefits of adaptive, codebook-free beam alignment at the UE side with the advantages of a codebook-sweep based scheme at the base station (BS). The proposed end-to-end trainable scheme is compatible with current cellular standard signalling and can be readily integrated into the standard without requiring significant changes to it.

The last repository presents implementation of an AI method for wide beam codebook design. The solution allows for exploration of the benefits of refined beam prediction and estimation. A neural network is applied as a decoder, where the measurement of signal powers acquired from the designed wide beams is the input, and the output is an estimate of the best UE refined beam. With the proposed idea, one can leverage the wide beam measurements from P1 to predict the refined beam having the highest reference signal received power (RSRP) for data transmission without needing the overhead from the P2 beam refinement procedure.

Each of the five repositories in this MIMO AI toolset includes detailed descriptions covering the problem solved, background, simulation environment/system models, functionality, and usage examples with results. These repositories offer valuable resources for researchers and developers, providing AI-based algorithms developed for varying physical layer challenges. In addition to allowing reproducibility of CENTRIC's research outputs, these repositories will serve as excellent starting points for exploring alternative solutions, benchmarking performance, and enabling advances in physical layer methods for the AI native Air Interface (AI-AI) concept. The repositories support integration with new AI algorithms and/or simulation environments thereby providing avenues for advancing research and development in this area.

List of authors

Company	Author	Contribution
Aalborg University	Ramoni Adeogun	Editor, WP3 lead and project TM
Aalborg University	Uyoata E. Uyoata	Contributor
Aalborg University	Xiyu Wang	Contributor
NVIDIA	Sebastian Cammerer	Contributor
NVIDIA	Jakob Hoydis	Contributor
Interdigital	Ahmet Serdar Tan	Contributor
Interdigital	Anouar Yatribi	Contributor
NSN	Amir Ahmadian Tehrani	Contributor
KEYSIGHT	Carles Navarro Manchon	Reviewer
EURESCOM	Halid Hrasnica	Reviewer and project coordinator

Table of Contents

List of figures and tables

List of figures:

List of tables:

Abbreviations

5G/6G	5th Generation/6th Generation
5G NR	Fifth Generation New Radio
AI	Artificial Intelligence/Air Interface
BM	Beam Management
CDL	Cluster Delay Line
CNN	Convolutional Neural Network
GNN	Graph Neural Network
ISAC	Integrated Sensing and Communication
LDPC	Low Density Parity Check Code
MCS	Modulation and Coding Scheme
MIMO	Multiple Input Multiple Output
NN	Neural network
NRX	Neural Receiver
PPO	Proximal Policy Optimization
PUSCH	Physical Uplink Shared CHannel
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SNR	Signal to Noise Ratio
TDD	Time Division Duplexing
TDL	Tap Delay Line
TDMA	Time Division Multiple Access

1 Introduction

CENTRIC's deliverable D3.4 is an open-source software libraries containing a selection of simulators of AI algorithms and associated simulation environments. The published open-source software library referred to as "CENTRIC's AI-based MIMO Toolset" contain five different repositories each with simulation codes and documentation of novel AI algorithms developed for the different components of the physical layer of the AI native Air-Interface (AI-AI). As shown in Table 1, the five repositories contribute to the four WP3 objectives.

Table 1: Mapping of WP3 Objectives to the AI algorithms and simulation environment published in the open-source software libraries

		WP3 Objectives			
		1. To devise AI-models and training procedures capable of providing end-to-end performance optimized waveforms	2. To develop and assess AI-methods for environment-specific beam management	3. To provide AI-enhanced MIMO processing solutions	4. To define and maintain a set of common scenarios and benchmark references for validation
Open-source repositories	RL Based Beam Management in ISAC Scenarios		✓		✓
	Multiuser MIMO Neural Receiver	✓		✓	✓
	Transfer Learning Techniques for Neural Receivers			✓	
	Learning Based Beam Alignment	✓	✓		✓
	Narrow Beam Prediction Using NN Decoder		✓		✓

The purpose of the open-source repositories, which contain simulators of AI algorithms and associated simulation environments, is to provide a comprehensive toolkit for researchers, developers, and engineers working on the development and performance evaluation of AI-based physical layer technologies, specifically within the AI-native Air-Interface (AI-AI) framework. These simulators allow users to evaluate and validate the novel AI algorithms developed within CENTRIC's WP3 on AI-AI physical layer methods. The goal is to facilitate experimentation with advanced techniques in MIMO, integrated sensing and communication (ISAC), beam management, alignment and prediction, and other key physical layer aspects of AI-driven communication systems.

By offering this open-source software library, CENTRIC encourages collaboration and innovation, enabling the broader research community to build upon the foundational work already done in the project. This openness will accelerate the development of 6G wireless communication systems by allowing for the replication of results, the creation of new AI models, and the extension of the published algorithms. The repositories provide not only the simulation codes but also detailed documentation to guide users in applying the tools effectively, fostering a shared knowledge base and promoting the adoption of AI-based solutions in wireless networks.

This report presents a description of the published repositories, detailing various key aspects that contribute to their development and application. Each repository is introduced with background information to contextualize the problem it addresses, highlighting the specific challenges in the AI-native Air-Interface (AI-AI) framework that the proposed AI algorithms aim to solve. It also outlines the simulation environment and system model used to develop and evaluate the performance of the algorithms. Furthermore, the report provides a description of the AI algorithms proposed within each repository, offering insights into their functionality and innovation. Additionally, usage examples are included to demonstrate practical applications of the repositories, guiding users through how to use and benefit from the tools. Lastly, the report present example results obtained from these simulators.

2 Repository 1: RL Based Beam Management in ISAC Scenarios

https://github.com/CENTRIC-WP3/RL-Based-Beam-Management-in-ISAC-Scenarios

2.1 Background

Beamforming and resource allocation methods have significantly improved because of the rising demand for fast and dependable wireless communication networks. A crucial task in wireless communication systems is beam alignment, which entails directing antenna arrays to maximize throughput and improve signal-to-noise ratio (SNR). Traditional beam sweeping in 5G allocates sensing resources for all users at fixed-slots periodicity. The operations are not typically adapted at short timescales, and the division between uplink (UL) and downlink (DL) slots is usually kept fixed over long time frames. This might lead to inefficient (low-mobility scenarios) and insufficient (high-mobility) sensing to track the necessity of beam changes. To address this, we aim to find a way to decide the allocation of resources between sensing, uplink, and downlink transmissions that is adaptive on a short timescale according to the network situation. We propose a Proximal Policy Optimization (PPO) solution for joint resource allocation and beam tracking in a millimeter-wave (mmWave) wireless communication context. By coordinating the optimization of the resource allocation and beam tracking parameters, our system intends to mitigate the mmWave channel's dynamic nature and its fluctuating channel quality, including blockage events. Simulation results show that, compared to baseline approaches, the proposed method outperforms them in terms of average packet error rate (PER), learning a more dynamic policy of slots allocation for efficient beam tracking and data transmission.

2.2 System Model

Consider a vehicular wireless network as depicted in Figure 1, using a time-division duplexing (TDD) scheme. A mmWave Base Station (BS) equipped with a uniform linear array (ULA) with

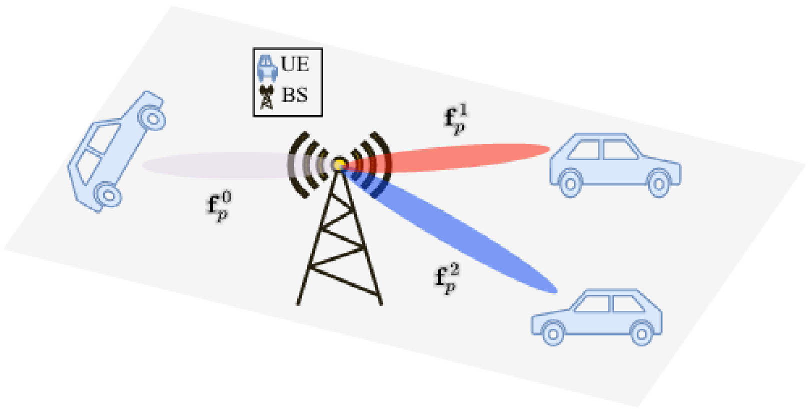

Figure 1: System model for vehicular wireless network

M_t transmitting antennas communicates with U single-antenna User Equipment (UEs) in a time-slotted network with $K = \{1, \dots, K\}$ orthogonal channel uses per frame. In each time slot, only one user u can be scheduled. Each user u has a buffer of size B_{UL}^u for uplink packets, while the base station has U buffers (with size B_{DL}^{u} for the user u) for downlink traffic. At each slot $k \in K$, the BS performs one action: S_u (sensing/beam sweeping), DL_u (transmission of downlink packet), or UL_u (reception of uplink packet) with $u = 1,\dots,U$.

2.2.1 Channel Model

Due to the high directivity of mmWave transmissions, we assume a geometrical two-state Line-of-Sight (LoS)/NLoS channel model for this work. In this way, the narrowband channel for the user u is given by $\mathbf{h}_u \in \mathbb{C}^{M_t}$

$$\mathbf{h}_{\boldsymbol{u}} = \frac{\sqrt{M_t}}{d_u} \beta_u \eta_u \boldsymbol{a}_t{}^{\dagger}(\theta_u),$$

where † denotes conjugate transposition, d_u denotes the distance between the BS and the u–th user, $\beta_u \sim CN(0, \sigma_u^2)$ is the complex fading gain with variance σ_u^2, and $\eta_u \in \{v_u, 1\}$ is the shadowing gain drop coefficient. When the user is in a LoS state $\eta_u = 1$, and $\eta_u = v_u$ otherwise. Also, θ_u denotes the Angle of Departure (AoD) with respect to the BS array axis and the u user position. The ideal isotropic array responses are given by

$$\boldsymbol{a}_t(\theta_u) = \frac{1}{\sqrt{Mt}} \left[1, e^{-j\pi\cos(\theta_u)}, \dots, e^{-j\pi(M_t-1)\cos(\theta_u)}\right]^T.$$

For the dynamics of the system, consider the user position in a given time slot $c_k^u = [x_k^u, y_k^u]$, its velocity $\mathbf{v} = [v_{linear}, v_{angular}]$, where $v_{linear} \sim \text{Exp}(1)$ and $v_{angular} \sim N(0,1)$ and a slot duration of Δt.This will change the AoDs and consequently make the channel vary along our time-slotted resources. We consider a 2D geometric layout.

2.2.2 Two-State Blockage Model

To have a dynamic scenario in which the users experience blockages, each user can be in a LoS/NLoS state within a time slot k. It follows a Bernoulli process with probability P_{NLOS}^{u}: $p_{NLOS}^{u} \sim \text{Bernoulli}(P_{NLOS}^{u})$, where $p_{NLOS}^{u} = \{0,1\}$ denotes a NLoS/LoS state. When in an NLoS state, we assume the duration for which the user is blocked lasts K_{NLOS} consecutive time slots. The blockage duration is modeled as a uniform distribution $K_{NLOS} \sim U(1, k_{NLOS})$.

2.2.3 Beam Codebook

A codebook-based analog beamforming architecture to beamform signals with a single RF chain at the BS is assumed. Let $\mathcal{F} = \{\mathbf{f}_1,\dots,\mathbf{f}_{M_t}\}$ be the codebook. We use the common Discrete Fourier Transform (DFT)-based codebooks, with precoders $\mathbf{f}_i$ given by

$$\mathbf{f}_i = \frac{1}{\sqrt{M_t}} \left[1, e^{-j\pi\frac{2i-1-M_t}{M_t}}, \dots, e^{-j\pi(M_t-1)\frac{2i-1-M_t}{M_t}}\right]^T, i \in \{1,\dots,M_t\}.$$

2.2.4 Traffic Model

Each user's traffic follows an independent Bernoulli process with probability P_u for generating packets in downlink and uplink in every time slot k, that is, $[p_{dl}^u, p_{dl}^u] \sim \text{Bernoulli}(P_u)$, where p_{dl}^u, p_{dl}^u takes values 0 or 1 to determine if a packet was generated for the user u in downlink and uplink for a time slot k.

2.2.5 Optimization Problem

The target is to minimize the average PER considering allocating resources for all users:

$$\min_{k\in\mathcal{K}} \frac{1}{UK}\sum_{u=1}^{U}\sum_{k=1}^{K} PER_{uk}$$

$$\text{s.t.} \sum_{u=1}^{U} s_{uk} + \sum_{u=1}^{U} d_{uk} + \sum_{u=1}^{U} u_{uk} = 1\,, \forall k = 1,2,\ldots,K$$

$$s_{uk}, d_{uk}, u_{uk} \in \{0,1\}, \forall u = 1,2,\ldots,U\,, k = 1,2,\ldots,K$$

where the constraints denote that just one user can be allocated in every slot k for sensing s_{uk}, downlink d_{uk}, or uplink u_{uk}, respectively.

2.3 AI Algorithm

PPO is utilized to solve the resource management problem. It is an actor-critic algorithm that utilizes neural networks to model both the policy and value functions, represented by parameters θ and ϕ respectively. The goal is to learn a policy $\pi_\theta(a|s)$ that maximizes the expected cumulative reward in an environment. At each time step k, the agent observes the current state s_k, selects an action a_k from the policy distribution $\pi_\theta(a_k|s_k)$, and receives a reward r_k and new state s_{k+1} upon executing the action. The agent stores these experiences to update its policy and value functions after accumulating a certain number of experience tuples. The advantage function A_k quantifies the advantage of taking action a_k in state s_k compared to the expected value from the current state, computed as the sum of discounted future rewards minus the value function at the current state:

$$A_k = \sum_{i=k}^{N} \gamma^{i-k} r_i - V_\phi(s_k)\,,$$

where N is the maximum number of time steps per experience memory. Next, the value function $V_\phi(s_k)$ estimates the expected cumulative reward from the current state s_k onwards. It is updated by minimizing the mean squared error between the estimated value and the target value:

$$L_{\text{critic}}^{(k)}(\phi) = \frac{1}{2}\Big(V_\phi(s_k) - \big(r_k + \gamma V_\phi(s_k + 1)\big)\Big)^2\,,$$

where γ is the discount factor that balances immediate and future rewards. The policy is updated using the PPO objective, which aims to maximize the expected advantage while avoiding large policy changes. The PPO loss function for a single time step is given by:

$$L_{\text{PPO}}^{(k)}(\theta) = \min\left(r_k(\theta)A_k, \text{clip}(r_k(\theta), 1 - \epsilon, 1 + \epsilon)A_k\right),$$

where the clip operation restricts the values from 1−ϵ to 1+ϵ, $r_k(\theta) = \frac{\pi_\theta(a_k|s_k)}{\pi_{\text{old}}(a_k|s_k)}$ is the ratio of the updated policy probabilities to the old policy probabilities, and ϵ is a hyperparameter that controls the magnitude of the policy change. To further improve the stability of training, an entropy regularization term is included. The entropy $S[\pi_\theta(a_k|s_k)]$ measures the uncertainty or randomness of the policy distribution $\pi_\theta(a|s)$.

This term encourages exploration by discouraging overly deterministic policies. Finally, the total loss function for PPO combines the policy loss, value function loss, and entropy regularization over the entire N experiences:

$$L_{\text{total}}(\theta,\phi) = \frac{1}{N}\sum_{k=1}(L_{PPO}^{(k)}(\theta) \; - \; c_1 L_{critic}^{(k)}(\phi) \; + \; c_2 S[\pi_\theta(a_k|s_k)])$$

where c_1 and c_2 are hyperparameters that control the trade-off between the value function loss and the entropy regularization. The policy and value networks are then updated jointly by minimizing the total loss.

2.4 Usage Example and Results

2.4.1 Usage Example

A PPO-based time slot allocation for minimizing PER is presented while the method here can be applied to the selection of any wireless resource. The BS, as the agent, takes beam indexes, packets in the DL and UL buffers, and the received power for the selected beams as the state:

$$\cdot\, k \; = \; \left\{\{i_1,\dots,i_U\},\{P_1^{DL},\dots,P_U^{DL}\},\{P_1^{UL},\dots,P_U^{UL}\},\{R_1^p,\dots,R_U^p\}\right\} \in \mathcal{S},$$

The action space is the set of all possible actions that the BS agent can choose from at each time slot. In this way, the possible actions are a discrete variable with $U \times 3$ possible values

$$a_k \in \mathcal{A} \; = \; \{(a,u), a \; = \; 0,1,2, u \; = \; 1,2,\dots,U\},$$

with the first element a denoting the action (0 for sensing, 1 for UL transmission, 2 for DL transmission) and the second element u denoting the target user. Once in a sensing slot s_{uk}, we assume we select the best neighboring beam (beam tracking) such that

$$\arg \max_{\mathbf{f}_i^u \in \{\mathbf{f}_{i-1}^u, \mathbf{f}_i^u, \mathbf{f}_{i+1}^u\}} \left|\sqrt{P_t}\, \mathbf{h}_u^T\, \mathbf{f}_i^u\, s \; + \; n_u\right|^2$$

where $\left|\sqrt{P_t}\mathbf{h}_u^T\mathbf{f}_i^u\, s \; + \; \mathbf{n}_u\right|^2$ is the received signal power of the u-th user, P_t, $s \in \mathbb{C}$, and $\mathbf{n}_u$ denote the transmission power, the known training symbol with normalized power, and the zero mean complex Gaussian noise vector with variance σ_n^2, respectively. If the sensing variable s_{uk} is active (i.e., s_{uk} = 1), the beam selection for either downlink (d_{uk}) or uplink transmission (u_{uk}) in slots will be based on the beam selected during the last sensing slot.

Packets can be dropped due to different reasons: either encountering a full buffer ($P_{DL}^u \; > \; B_{DL}^u$ or $P_{UL}^u > \; B_{DL}^u$), transmitting with a poor quality beam (i.e., using an outdated beam $\mathbf{f}_i^u$ obtained from equation (15) during s_{uk}), or transmitting during a blockage (p_{NLOS}^u = 1). A beam is considered outdated if $\mathbf{f}_i^u$ is no longer the optimal beam. Reward is defined accordingly as:

- $G_b^u(k)$ acts as an indicator of buffer fullness for user u at time step k. $G_B^u(k) \; = \; 1$ is given when the buffer is not full.
- $\rho(k)$ captures the impact of beam tracking on beam quality. Positive values encourage adaptive beam tracking, while negative values discourage unnecessary adjustments.
- $O(k)$ serves as a penalization factor for drops induced by blockages.

- $D(k)$ represents the number of packets dropped within time slot k. Furthermore, the reward r_k is defined as

$$r_k = \frac{1}{U \times 2} \sum_{u=1}^{U} \sum_{b \in \{DL,UL\}} G_b^u(k) + \rho(k) + O(k) - D(k)$$

2.4.2 Results

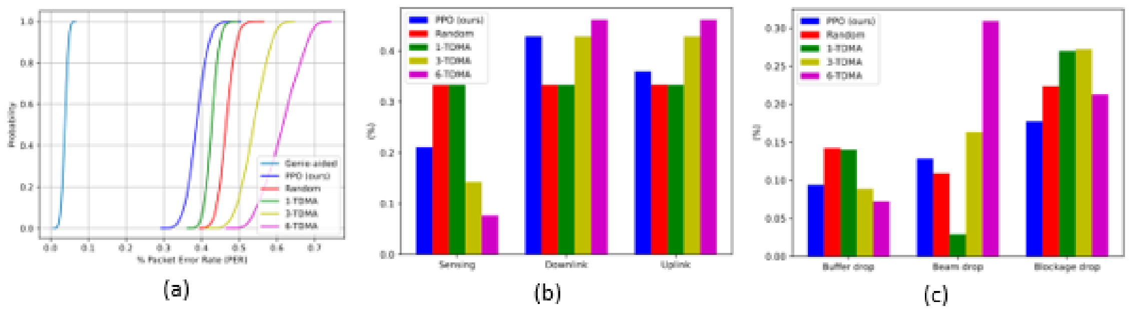

Figure 2: Simulation results over test episodes: (a) ECDF of the average PER, (b) slots distribution, and (c) drops distribution.

The benchmark and baselines to evaluate the results are given below: 1) *Genie-aided (benchmark)*: This agent knows exactly when blockages are occurring and always has the best beam for the user in a TDMA fashion. This will be the benchmark for the proposed solution. 2) *Random*: The policy of the agent is to allocate slots uniformly at random among all possible actions. 3) *X-TDMA* (X Time Division Multiple Access) is a slot allocation policy where slots are divided into sensing (S), uplink (UL), and downlink (DL) categories. The value of X represents the number of consecutive UL/DL slots pair before transitioning to the next sensing slot. These patterns are allocated for every user u in a round-robin fashion.

Figure 2 shows the empirical cumulative distribution function (ECDF) of the average PER, the slot distribution and packet drop distribution over the test episodes, respectively. The genie-aided approach exhibits an effective benchmarking of the PER. Compared to random and X-TDMA slot allocation, PPO method minimizes the average PER and finds a balance between sensing and communication slots. Excessive slot allocation for DL/UL compromises results, as evidenced by a performance degradation of around 40% in 3/6-TDMA. Furthermore, 6-TDMA shows high variance in PER results due to scenario variation and a lack of sensing slots. Although reducing the number of sensing slots, the PPO agent still reduces the buffer drop against 3/6-TDMA. Moreover, PPO decreases significantly buffer drops in comparison with 1-TDMA, but also places importance on the two other drop cases. However, as 1-TDMA fixed slot allocation is more robust in terms of beam drops, it does not effectively adapt to the dynamics of the environment as its overall PER performance is worse than our solution. To sum up, the results demonstrate that the agent learns a policy of a dynamic slot allocation to both enhance performance and reduce resource allocation wastage.

2.5 Remarks

The results of the exemplified PPO-based time-slot allocation demonstrate that the model can successfully learn the complicated dynamics of the environment and provides an effective solution to the resource allocation problem. The model exhibits great generalization performance of allocation of other resources regardless of system stochasticity. The approach is promising for integrated sensing and communications networks.

3 Repository 2: Multiuser MIMO Neural Receiver

https://github.com/CENTRIC-WP3/neural_rx

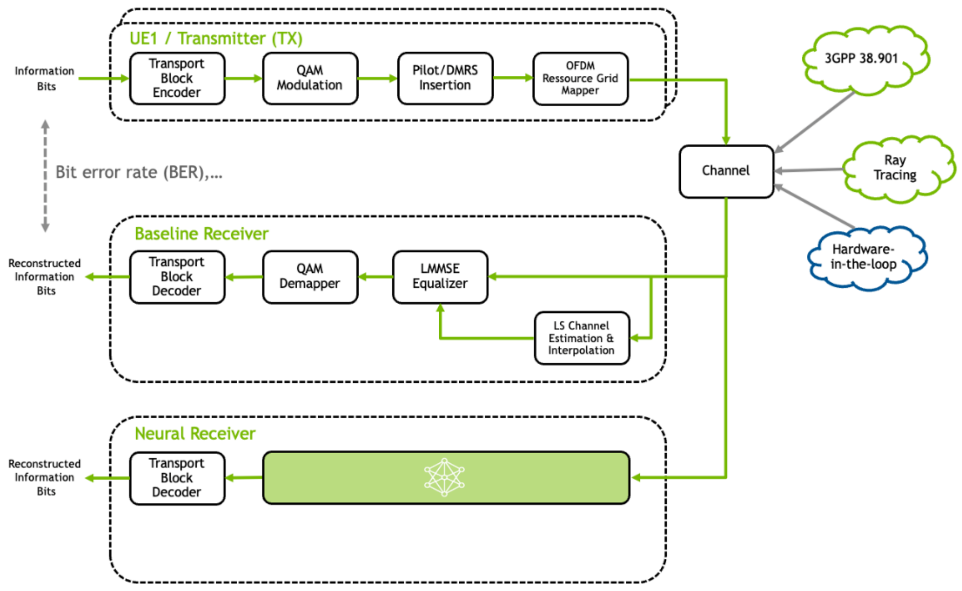

Figure 3: A neural receiver replaces channel estimation, equalization and demapping by a single neural network.

As part of the CENTRIC project, the consortium explores neural network (NN)-based receivers that comply with 5G New Radio (5G NR), positioning them as enabling technology for novel applications such as re-trainable site-specific base stations and pilotless communications. Initially proposed for single-input multiple-output (SIMO) systems, NN-based receivers have since evolved to support multiple-input multiple-output (MIMO) systems and pilotless communication setups. We have extended this concept as part of WP3 to a flexible multi-user MIMO (MU-MIMO) receiver with 5G NR physical uplink shared channel (PUSCH) compatibility. The receiver architecture combines graph neural networks (GNN) and convolutional neural networks (CNN), allowing flexibility in handling varying user numbers and sub-carrier configurations without retraining.

Additionally, we detail the steps required to deploy a MU-MIMO neural receiver (NRX) in actual cellular communication systems, addressing challenges such as real-time inference and 5G NR standard compatibility. The developed NRX architecture can support dynamic modulation and coding scheme (MCS) configurations without retraining and achieving inference times of less than 1ms on an NVIDIA A100 GPU using the NVIDIA TensorRT inference engine. The architecture is optimized to minimize the signal-to-noise ratio (SNR) performance degradation, and we explore site-specific adaptation for enhancing performance in different radio environments. The resulting NRX is ready for deployment in real-time 5G NR testbeds, and the released source code includes the TensorRT experiments and pre-trained weights.

Please note that, the neural receiver architecture and its real-time optimization has been already described in deliverable D3.2, D3.3 and D2.1. Thus, we keep the introduction of the algorithm short and refer the interested reader to these deliverables.

3.1 Background

One of the key objectives of the CENTRIC project is the exploration and development of AI-enabled, user-centric communication systems capable of adapting to new situations and applications. On the physical layer, this involves introducing trainable components and trainable parameters into signal processing algorithms that can be adjusted through training. A particularly promising approach is the so-called neural receiver (NRX), which replaces significant portions of traditional physical layer receiver algorithms with neural networks, as illustrated in Figure 3. The NRX is not merely a *drop-in replacement* for existing receiver algorithms but serves as a foundational technology for enabling a plethora of novel features, such as site-specific fine-tuning of the receiver and pilotless communications via end-to-end learning.

To date, most studies have been simulation-based, and the real-time inference latency implications of the proposed solutions remain largely unexplored. The stringent latency and throughput requirements of wireless communication systems impose strict constraints on neural network (NN) design, limiting their size and complexity. Therefore, deploying and validating AI/ML components in the physical layer of a real cellular system under realistic latency conditions presents an open and intriguing challenge. Addressing this challenge is a key goal of the CENTRIC consortium, as it seeks to demonstrate the practical viability of such AI/ML algorithms in a real hardware-in-the-loop testbed.

3.2 Simulation Environment/System Model

The code in this repository allows to design, train, and evaluate neural receivers [1] using the NVIDIA® Sionna™ link-level simulation library and TensorFlow. Further, trained receivers can be prepared for real-time inference via NVIDIA® TensorRT™.

The following features are currently supported:

- 5G NR compliant Multi-user MIMO PUSCH receiver
- Training pipeline using 3GPP compliant channel models
- TensorRT / ONNX model export for real-time inference
- Support for varying number of PRBs, users, and different MCS schemes per user
- End-to-end learning of custom constellations for pilotless communications
- Site-specific training using ray-tracing based channel simulations from SionnaRT

We recommend starting with the *Jumpstart NRX Tutorial* notebook for a detailed introduction and overview of the project.

The basic neural receiver architecture is introduced and described in [1]. The real-time experiments and the site-specific training is described in [2].

Running this code requires Sionna 0.18. To run the notebooks on your machine, you also need Jupyter. We recommend Ubuntu 22.04, Python 3.10, and TensorFlow 2.15. For TensorRT, we recommend version 9.6 and newer.

3.3 AI Algorithm

As mentioned above, a detailed NRX introduction can be found in deliverables D2.1, D3.2 and D3.3. However, we briefly recap the NN architecture in the following. The architecture was introduced in [1] and consists of the following elements:

- Convolutional neural network (CNN) layers over the time-frequency grid
- Graph neural network (GNN) inspired multi-user interference cancellation scheme
- Readout network to project the feature space back to the desired output domain (LLRs and channel estimates)

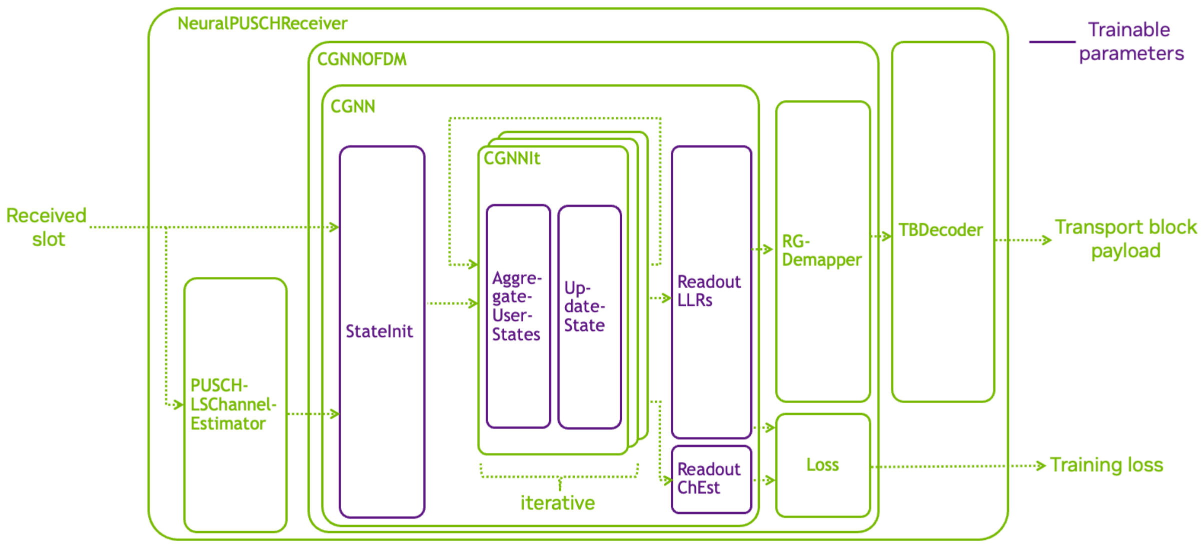

Figure 4: Overview of the NeuralPUSCHReceiver integrated as Keras layer in the NVIDIA Sionna link-level simulator.

An overview of the network layers is given in Figure 4. Please note that a few minor steps are ignored to simplify the visualization. The network implements a similar functionality as the Sionna PUSCHReceiver and directly returns the reconstructed payload bits of the transport block. However, the LDPC decoder and rate-matching is done by a *classical* transport block decoder and is, thus, not trainable.

The detailed receiver algorithm is described in [1] and can be summarized as follows.

The NRX takes as input:

- Received signal of slot
- Initial channel estimate for each user (e.g., from least square (LS) channel estimation)

- Positional encoded pilot positions in time/frequency grid

The receiver (*CGNN*) consists of the following stages:

1. Initial convolutional neural network (CNN) that projects the stacked input to a high dimension feature space (*StateInit*)
2. Pointwise MLP and aggregation between the resource elements (REs) of all users acting as multi-user interference cancellation (*AggregateUserStates*)
3. CNN over the time/frequency grid. This is implemented with separable convolutions and skip connections (*UpdateStates*)
4. Readout MLP to project the internal state back to scalar LLRs (*ReadoutLLRs* and *ReadoutChEst*)

Steps 2 and 3 are iteratively repeated, which effectively defines the depth of the receiver and, thereby, enables a simple computational complexity adjustment.

Outer components (*CGNNOFDM*) such as the initial channel estimator (*PUSCHLSChannelEstimator*), the resourcegrid demaper (*RGDemapper*) and the transport block decoder (*TBDecoder*) are non trainable components, but required for 5G NR compatibility. We use the standard components from the Sionna link-level simulator.

3.4 Usage Example and Results

This repository is structured in the following way:

- *config* contains the system configurations for different experiments
- *notebooks* contains tutorials and code examples
- *scripts* contains the scripts to train, evaluate and debug the NRX
- *utils* contains the NRX definition and all Python utilities
- *weights* contains weights of pre-trained neural receivers for different configuration files
- *results* contains pre-computed BLER performance results

We recommend starting with the *Jumpstart NRX Tutorial* notebook for a detailed introduction and overview of the project.

The workflow consists of the following steps:

1. Define a *config-file*: this includes the entire system configuration including detailed NRX architecture and training/evaluation parameters
2. *Train* the NRX and track training performance in Tensorboard
3. *Evaluate* the BLER performance of the receiver and compare against baselines

4. *Export* the model to ONNX fileformat and run as TensorRT-engine for real-time performance analysis

Furthermore, this framework also supports end-to-end learning for pilotless communications [2]. For further information, we refer to the *end-to-end learning* notebook in the repository.

After defining a config file, training can be done with the following command:

"python ../scripts/train_neural_rx.py -config_name CONFIG_NAME.cfg -gpu 0"

After training, the evaluation can be done with

“python evaluate.py -config_name CONFIG_NAME.cfg -gpu 0 -num_tx_eval 1”

Afterwards, the results can be visualized using the *plot results* notebook. And example of the performance after training is given in Figure 5. The large neural receiver uses approx. 4x more trainable parameters as the optimized real-time version. A performance degradation of approx. 0.7dB can be observed. However, the inference latency improves from approx. 3.11ms to 0.96ms. For further details on the inference latency, we refer to D2.1.

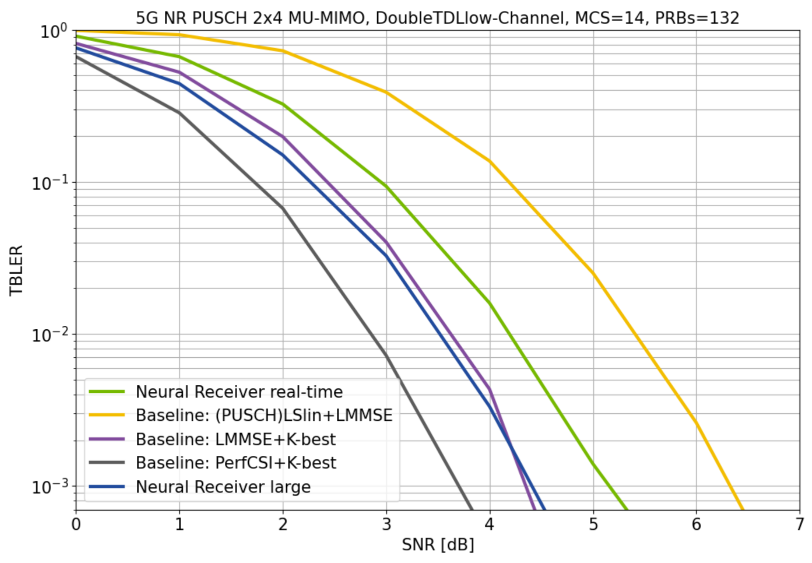

Figure 5: Block error rate performance of classical and AI/ML-based receiver algorithms. All receivers are evaluated over the same TDL-B/TDL-C channel models using 2 UEs and 4 receive antennas

3.5 Remarks

Parts of the content of this report will be made publicly available as academic publications.

4 Repository 3: Transfer Learning Techniques for Neural Receivers

https://github.com/CENTRIC-WP3/Transfer-Learning-Techniques-for-MIMO-Neural-Receivers

4.1 Background

Training deep neural network models typically requires large datasets. Due to the dynamic nature of wireless communication setups, generating sufficient dataset for each configuration of wireless communication deployment can be cumbersome. To overcome this hurdle, transfer learning can be exploited wherein the weights of a source model trained on a large dataset can be modified fully or partially by the relatively smaller dataset of the target model. For transfer learning to provide reasonable gains, there must be some relationship between the source dataset/task and the target dataset/task [3]. For this task we set out to evaluate the performance of some transfer learning techniques for dep neural SIMO receiver. To this end we consider two fine tuning techniques and a feature extraction technique. We consider cases of subcarrier spacing and channel model mismatch and compare the performance of the transfer learning techniques with some benchmark approaches. We further validate the performance of the transfer learning approach using static data set.

4.2 Simulation Environment/System Model

An end-to-end communication chain is considered where at the transmitting end, bits are generated, encoded, mapped to constellations, and positioned on physical resource blocks. OFDM symbols resulting from IFFT are filtered through 3GPP channel models. AWGN is added at the receiver and the DFT transformed symbols are fed into a neural receiver which outputs the Log Likelihood Ratios of the received symbols. Details of the neural receiver architecture are given in Table 2, and in [4].

Table 2: Neural Network Architecture

Layer	**Channels**	**Kernel Size**	**Dilation Rate**
Input Conv2D	128	(3,3)	(1,1)
ResNet 1	256	(3,3)	(1,1)
ResNet 2	256	(3,3)	(1,1)
ResNet 3	256	(3,3)	(1,1)
ResNet 4	256	(3,3)	(1,1)
Output Conv2D	Number of bits/symbols	(3,3)	(1,1)

4.3 AI Algorithm

We evaluate 2 finetuning based algorithms which we call *finetuning* and *finetuning +.* In finetuning, the architecture of both the source neural receiver and the target neural receivers are the same. All the weights transferred from the source receiver are modified using data derived from the target. In finetuning + the architecture of the target neural receiver has an additional ResNet block, and the top two layers of the target model are frozen hence not modified by target dataset. Similarly, the feature extraction technique has the same architecture as in the finetuning + case, however only the last two layers of the target model are modified with target dataset with the rest layers frozen. A workflow of these techniques can be seen in Table 3 and Table 4.

Table 3: Flow of the Finetuning based techniques.

```
Algorithm 1: Technique 1 and 2: Fine Tuning
Input: Source deep neural receiver ($f_{\Phi_S}$), $k$, $N_{iterations}$
Output: BLER of target deep neural receiver, ($f_{\Phi_T}$)
If "Fine Tuning" then
      Retain $f_{\Phi_S}$ architecture
      Load source network parameter, $\Phi_T \leftarrow \Phi_S$
      for $i = 1$ to $N_{iterations}$ do
            Generate target dataset, $X_T$
            Input dataset into target network
            Update the parameters of $f_{\Phi_T}$ for trainable layers
      end
else
      If "Fine Tuning +" then
            Add an extra ResNet block
            Load source network parameters $\Phi_T \leftarrow \Phi_S$
            Freeze k layers of $f_{\Phi_T}$
            For $i = 1$ to $N_{iterations}$ do
                  Generate target dataset, $X_T$
                  Input dataset into target network
                  Update the parameters of $f_{\Phi_T}$ for trainable layers
            end
      end
end
Instantiate target model $f_{\Phi_T}$ and load weights, $\Phi_T$
Evaluate target model on test data
Output BLER of target model/ receiver
```

Table 4: Logical flow of the feature extraction technique

```
Algorithm 2: Feature Extraction
Input: Source deep neural receiver ($f_{\Phi_S}$), $k$, $N_{iterations}$
Output: BLER of target deep neural receiver, ($f_{\Phi_T}$)
Add an extra ResNet block
Load source network parameters $\Phi_T \leftarrow \Phi_S$
Freeze all layers of $f_S$
For $i = 1$ to $N_{iterations}$ do
            Generate target dataset, $X_T$
            Input dataset into target network
            Update the parameters of $f_{\Phi_T}$ for trainable layers
      end
Instantiate target model $f_{\Phi_T}$ and load weights, $\Phi_T$
Evaluate target model on test data
Output BLER of target model/ receiver
```

4.4 Usage Example and Results

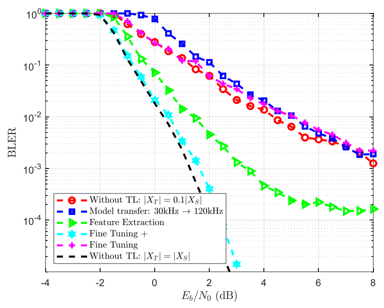

Figure 6: Figure 6: BLER Vs. Eb/No for Transfer between SCS 30kHz to 120kHz

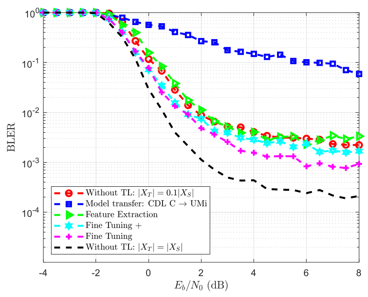

Figure 7: BLER Vs. Eb/No for Transfer between CDL C to 3GPP UMi

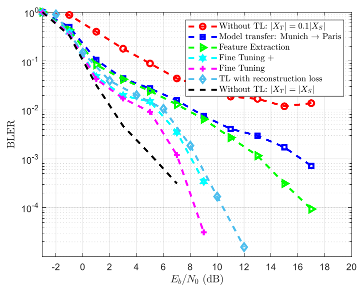

Figure 8: BLER Vs. Eb/No for Transfer between Munich and Paris Sites

We consider subcarrier spacing mismatch and channel model mismatch cases. In Figure 6, the CDL-C channel model is used, and subcarrier spacing mismatch is studied. Partially fine-tuning the weights from the source model offers improved target receiver performance even though the target dataset size is 10% of the size of the source dataset. On the other hand, in Figure 7, fully fine tuning the transferred weights offers improved target receiver performance for the channel model mismatch case. As the channel model plays a critical role in communication system performance, mismatch in the channel model between source and target model will require altering transferred source weights with target dataset. To obtain the results in 6 and 7, training was done on the fly. To validate the results, we considered transfer learning using raytracing generated site-specific static dataset. The source dataset was generated for a Munich site while the target dataset was generated for a location in Paris. Other configurations were kept same for both source and target receivers. With *Fine tuning,* the target receiver achieves a performance 2dB shy of the traditional deep learning approach. For Figure 8, we also compared our earlier studied techniques with a so-called transfer learning with reconstruction loss proposed in [4]. Although Transfer Learning with Reconstruction Loss performed nearly as *Finetuning+*, it was bested by *Finetuning*.

4.5 Remarks

Transfer learning for neural receivers is achievable for various mismatch scenarios using both on the fly data and static dataset. Finetuning based techniques can close the gap between having sufficient data and having no target domain dataset. Performance of transfer learning techniques for neural receivers is closely coupled with the mismatch cases. While partial finetuning is optimal in receiver configuration mismatch, full finetuning offers an edge in channel model mismatch.

5 Repository 4: Learning Based Beam Alignment

https://github.com/CENTRIC-WP3/Learning-Based-Beam-Alignment

5.1 Background

The challenging propagation environment, combined with the hardware limitations of mmWave systems, gives rise to the need for accurate initial access beam alignment strategies with low latency and high achievable beamforming gain. Much of the recent work in this area either focuses on using beam-codebooks together with methods like compressed-sensing, machine learning, and Bayesian approaches [5] [6]. It has been shown that codebook-free, adaptive beam alignment might have performance benefits over non-adaptive and codebook-based approaches [7], and some studies further showed the benefits of joint two-sided schemes [8]. This study introduces a novel deep learning based joint two-sided beam alignment scheme that aims to combine the benefits of adaptive, codebook-free beam alignment at the UE side with the advantages of a codebook-sweep based scheme at the base station (BS). The proposed end-to-end trainable scheme is compatible with current cellular standard signaling and can be readily integrated into the standard without requiring significant changes to it. Extensive simulations demonstrate superior performance of the proposed approach over purely codebook-based ones.

5.2 Simulation Environment/System Model

We consider the problem of joint, two-sided beam alignment in mmWave communications, i.e., the initial (downlink) communication between a BS and a UE, both equipped with an uniform linear array (ULA) consisting of N_{TX} and N_{RX} antenna elements respectively. No initial knowledge about the channel between BS and UE is assumed. We assume that both the BS and the UE are controlled by AIML-based control units which at each timestep t, that determine the current precoding vector $f_t \in \mathbb{C}^{N_{TX}}$ and the combining vector $w_t \in \mathbb{C}^{N_{RX}}$ at the BS and UE respectively, for the channel matrix $\mathbf{H} \in \mathbb{C}^{N_{RX} \times N_{TX}}$. Also, the control unit at the UE also receives the complex valued received symbol $y_t \in \mathbb{C}$ and the current BS beam index $y_t \in \mathbb{Z}$ at each timestep as input. The system model is depicted in Figure 9.

Simulation environment is Python based and use PyTorch as the machine learning library.

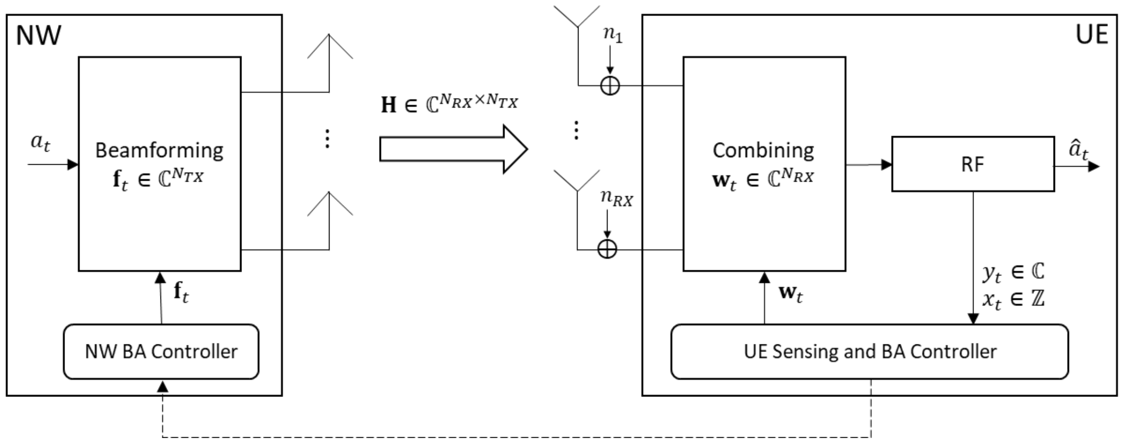
NW
a_t
Beamforming
$\mathbf{f}_t \in \mathbb{C}^{N_{TX}}$
$\mathbf{f}_t$
NW BA Controller
$\mathbf{H} \in \mathbb{C}^{N_{RX} \times N_{TX}}$
n_1
n_{RX}
UE
Combining
$\mathbf{w}_t \in \mathbb{C}^{N_{RX}}$
RF
$\hat{a}_t$
$y_t \in \mathbb{C}$
$x_t \in \mathbb{Z}$
$\mathbf{w}_t$
UE Sensing and BA Controller

Figure 9: System model for learning based beam alignment

5.3 AI Algorithm

The unrolled depiction of the proposed algorithm is provided in Figure 10. We use three different kinds of learnable networks: The UE sided recurrent neural network (RNN) (N1), the final beam-mapping network at the BS (N2), and the learnable beam-codebook at the BS (N3).

N1 (RNN): Sensing network with the goal of obtaining as much CSI information as possible about the channel between BS and UE, based on the measurement sequence of w_t, y_t and x_t. The obtained information is stored in the internal state s_t. At each timestep t, it outputs a combining vector w_t, and for $t = T - 2$, it additionally determines a feedback massage m_{FB} for the BS. Also, at the last timestep of the BA process, it outputs the estimated best combining vector w_{T-1} for the UE.

N2 (feedforward neural network (FNN)): Network to map the feedback massage m_{FB} of N1 to the final beampattern f_{T-1} at the BS.

N3 (parameter matrix): Learnable beam-codebook at the BS for possible enhancements over classically used codebooks and hardware impairments. This codebook is implemented as a $2 \times N_{TX} \times N_{TR}$ dimensional trainable parameter matrix.

The objective of the proposed method is to find the parameters of the neural networks N1, N2, and N3 that maximize the following function on beam forming gain at the last time step.

$$\text{Beamforming gain} = \frac{||w_{T-1}^H \mathbf{H} f_{T-1}||^2}{||\mathbf{H}||_2}$$

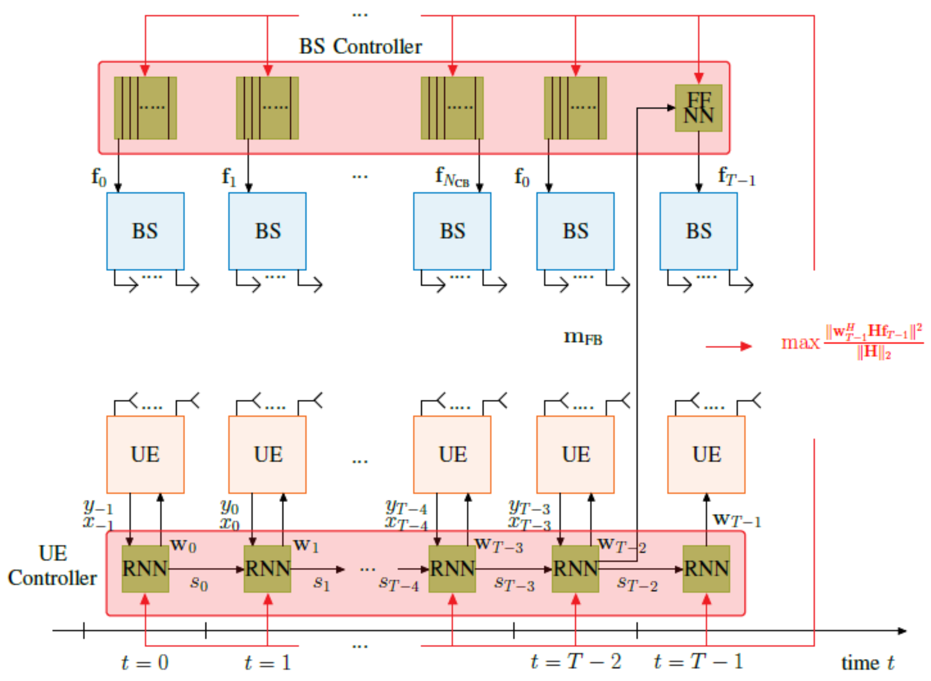

Figure 10: Unrolled depiction of the beam alignment algorithm

5.4 Usage Example and Results

See the README file in the repository for an example usage and obtained results. An example result is given in Figure 11 for the above parameters where C1, C2, C3 represents the results with N1, N2, N3 respectively. MRT+MRC represents the upper bound and Exhaustive search represents the legacy method of exhaustive search in 3GPP.

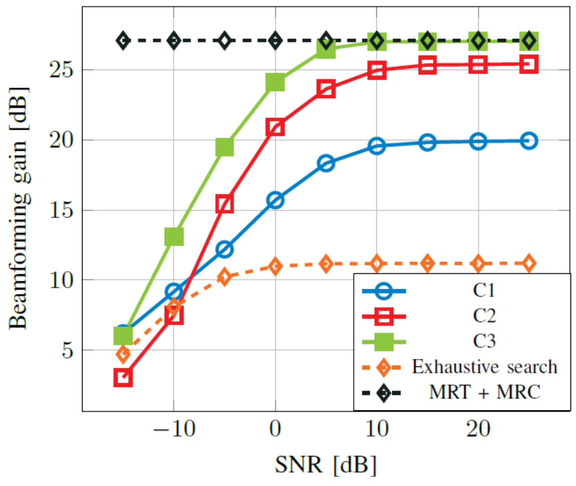

Figure 11: Comparison of beamforming gains of different methods

5.5 Remarks

This repository contain code for the unrolled AI algorithm described in Section 5.3. The codes allow for easy evaluation of the methods and benchmarking.

6 Repository 5: Narrow Beam Prediction Using NN Decoder

https://github.com/CENTRIC-WP3/Narrow-Beam-Prediction-using-Neural-Network-Decoder

6.1 Background

The procedure for acquiring and maintaining the highly directional beamforming with hybrid antenna arrays for data transmission at both the Next Generation Node B (gNB) and the UE is referred to as BM in 5G NR [9]. The BM procedure consists of three phases. In (P1), gNB sweeps all the transmitting (Tx) beams to broadcast synchronized signal blocks (SSB). The Tx beams in P1 are usually designed to have a relatively wide beamwidth to reduce the number required to fully scan the coverage area. In (P2), further Tx beam sweeping may be performed to select a preferred "refined" beam having a higher gain and narrower beamwidth so as to achieve a higher throughput during data transmission. The gNB may transmit channel state information reference signals (CSI-RS) over a set of refined beams, where the main lobes of the beams in the set of refined beams are all located within the beamwidth of the wide beam selected in P1. In (P3), the gNB uses the beam selected in P2 to transmit CSI-RS over successive time instances to allow the UE to switch between different receiving Rx beams so that the UE can determine the best Rx beam based on the received signal power. More recently, 3GPP has started to study applying AI/ML techniques to beam management in the NR air interface. ML based beam management can potentially improve legacy 5G beam management operation to reduce measurement overhead, communication latency, and device power consumption [10]. The 3GPP Release-18 Study Item has been supported by significant research advancements in the application of ML techniques to solve different problems for mmWave beamforming. To cite a few examples, [11] utilize neural networks (NNs) to design the sensing beam codebook and/or design the beam measurement decoder. The sensing beams essentially result from combining refined beams into wide beams, while the measurement decoder inputs the measurements of the wide beams and provides the estimation of certain channel state information. Considering the mmWave channel angle of departure (AoD) or angle of arrival (AoA) estimation, the AoD estimation with wide beam measurements as a super-resolution problem, may use convolutional NN (CNN) and long short-term memory (LSTM) as a wide beam measurement decoder or fully connected NN (FNN) for a wide beam measurement decoder [4]. Moreover, [5] proposes the use of autoencoder to jointly design the wide beam codebook and the wide beam measurement decoder. In the literature, the mmWave beamforming codebook design problem was also addressed with non-ML techniques. [12] proposes a wide beam construction method based only on phase coefficients and optimizes the combining coefficients to flatten the wide beam main lobe.

Motivated by recent standardization efforts, this study details a method for wide beam codebook design and shows that it is beneficial for refined beam prediction and estimation. We apply a neural network as a decoder, where the measurement signal powers acquired from the designed wide beams is the input, and the output is an estimate of the best UE refined beam. With the proposed idea, one can leverage the wide beam measurements from P1 to predict the refined beam having the highest reference signal received power (RSRP) for data transmission without needing the overhead from the P2 beam refinement procedure.

This study attempts to fill in the gap via formulating the problem of refined beam prediction with wide beam measurements as a classification problem by only utilizing the received signal power measurements as our prediction model input, which is more practical than assuming full knowledge of the complex received signals as presented in the prior-art. The main aspects of the study are summarized below:

- To propose an algorithm that can generate a wide beam codebook to maintain the cell serving coverage (not considered in the prior art) and improve the refined beam prediction accuracy.
- To propose to apply a NN with a fully connected layer and a residual connection to decode the wide beam measurement and predict the best refined beam.

6.2 Simulation Environment/System Model

6.2.1 System model

6.2.1.1 Refined beam problem formulation

Consider a gNB with a uniform planar array (UPA) having $N \times M$ antenna elements, where N represents the azimuth antenna elements and M represents the elevation antenna elements. The respective spacings for these elements in the azimuth and elevation directions are d_az and d_ele. The gNB coverage, mapped into its Tx beam angle domain, has an elevation angle range $[\theta_{\min}, \theta_{\max}]$ and an azimuth angle range $[\varphi_{\min}, \varphi_{\max}]$. The Kth gNB refined beam is given by:

$$\mathbf{v}_k = \mathbf{v}_{az}(\theta_m, \phi_n) \otimes \mathbf{v}_{ele}(\theta_m)$$

where v_{ele}denote the elevation beam given by

with

$$\mathbf{v}_{ele}(\theta_m) = \frac{1}{\sqrt{M}}[1, e^{\frac{j2\pi f_c d_{ele} \sin\theta_m}{c}}, \cdots, e^{\frac{j2\pi f_c d_{ele} \sin\theta_m (M-1)}{c}}$$

and

$$\mathbf{v}_{az}(\theta_m, \phi_n) = \frac{1}{\sqrt{N}}[1, e^{\frac{j2\pi f_c d_{az} \cos\theta_m \sin\phi_n}{c}}, \cdots, e^{\frac{j2\pi f_c d_{az} \cos\theta_m \sin\phi_n (N-1)}{c}}]^T,$$

Here, $M^{'}$and $N^{'}$denote the number of beams in azimuth and elevation. The refined beam index k is computed as $k = n + (m-1)\, N'$ and $k \in [1, M'N']$. The gNB refined beam codebook is expressed as:

$$\mathcal{B}_{refined} = \left[\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{M'N'}\right] \in \mathbb{C}^{NM \times N'M'}$$

6.2.1.2 Wide beam problem formulation

Since the beam sweeping process must periodically align the transmit beam with the updated UE position, using refined beams for the beam sweeping process incurs significant overhead from the large $N^{'}M$ beams. Consequently, the gNB might adopt an alternative transmit beam codebook with only $\frac{N^{'}M^{'}}{S}$ beams for the beam sweeping, where $S \geq 1$ is the refined beam combining ratio. The beam sweeping phase beam codebook mainly includes wide beams with larger beamwidths. The gNB wide beam codebook is defined as:

$$\mathcal{B}_{wide} = \left[\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_{\frac{N'M'}{S}}\right] \in \mathbb{C}^{NM \times \frac{N'M'}{S}}$$

where w_j is the wide beam vector. In this work, we aim to utilize the wide beam measurement power reported by the UE to the gNB to identify the optimal refined beam (i.e., the beam with the highest RSRP) for the UE.

6.2.1.3 Wide beam codebook design

Each wide beam vector S is calculated by linearly combining the refined beams, and in practice, such operation can be implemented with summation operation from the analog antenna chains without amplitude tuning. The beam vector w_i is expressed as:

$$\mathbf{w}_i = \frac{1}{\sqrt{S}} \sum_{k=\mathbf{k}^{((j-1)S+1)}}^{\mathbf{k}^{(jS)}} \mathbf{v}_k e^{j\omega_i k}$$

Where $k^{(k)}$is the kth element in k, and ω_i is phase coefficient for the linear refined beams combination. Adopting the wide beam beamwidth, ω_i can be calculated to flatten the wide beam main lobe.

Two main objectives for the wide beam codebook design can be listed as follows:

1) B_{wide} should satisfy the cell coverage conditions.
2) $B_{refined}$ is expected to enhance the refined beam prediction accuracy in comparison to the wide beam codebook generated.

Wide beam codebook B_{wide} generation steps are briefly summarized below:

- **Input:** refined beam codebook $B_{refined}$, combining rate S
- **Output:** wide beam codebook B_{wide}
- **Step 1:** determine matrix T and refined beam index *k*
- **Step 2:** construct all the wide beam vectors $\mathrm{w_i}$
- **Step 3:** return wide beam codebook B_{wide} = [$w_1,\ldots,w_{\frac{N'M'}{S}}$]

6.2.2 Simulation environment

The NN training data are generated from a 5G NR 3GPP-compliant system level simulator (SLS), and the specific configuration of this SLS is detailed in Table 5.

Table 5: System Level Simulation Parameter

Scenario	3GPP TR 38.901 UMa
Deployment	Hexagonal grid, 7 gNB sites, 3 sectors per site, 21 cells, gNB antenna height 25 m, (downtilt 13°), UE antenna height 1.5 m
Inter-cell distance	200 m
System bandwidth	80 MHz
Frame structure	TDD, DL data frame only
Carrier frequency	30 GHz
Sub-carrier spacing	120 kHz
gNB antenna	$(M = 4, N = 8, P = 2)$
gNB grid of beam config	32 transmit beams: 8 Azimuth angles (°) $= -56.25 + 15n, n = 0, \ldots, 7$ 4 Elevation angles (°) $= \{-6.6 - 19.8 - 33 - 46.2\}$ refined beam combining ratio $S = 4$
UE antenna	$(M = 1, N = 4, P = 2)$
HARQ	No re-transmissions
Traffic model	Full buffer
UE distributions	10 UEs per sector, random, 100% UE Outdoor
UE speed	3 km/h

The simulations involved 200 drops, encompassing a total of 42,000 UEs. The data destined for model training, validation, and testing are segmented with a splitting ratio of 0.8: 0.1: 0.1. The training of the model utilizes the Adam optimizer along with the StepLR learning rate scheduler, starting with an initial rate of 0.01. For model training, we use the binary cross-entropy loss function, expressed and RSRP is the ground-truth received signal power vector from all refined beams.

6.3 AI Algorithm

A neural network (NN) is employed as the wide beam measurement decoder to predict the indices of the optimal refined beams. With the introduction of NN, the objective function can be reformulated as

$$\hat{k}^* = arg\max_{k} [\mathfrak{y}_1, \cdots, \mathfrak{y}_k, \cdots, \mathfrak{y}_{M'N'}]$$

$$[\mathfrak{y}_1, \cdots, \mathfrak{y}_{M'N'}] = f(\bar{\mathrm{p}}, \Phi)$$

and f(.,Φ) is the neural network parameterized by by Φ and $\eta_k$represents k-th output of the last layer of the NN with input of received signal power sequence before normalization.

The architecture of the proposed NN decoder is depicted in Table 6.

Table 6: Neural Network Decoder Architecture

Layer	Structure
1	LayerNormalization
2	LayerNormalization (ReLU (Dense $\left(\frac{M'N'}{S}, M'N'\right)$))
3	LayerNormalization (ReLU (Dense $(M'N', M'N')$))
4	LayerNormalization (ReLU (Dense $(M'N', 2M'N')$))
5	Dropout (0.5)
6	LayerNormalization (ReLU (Dense $(2M'N', M'N')$))
7	LayerNormalization (ReLU (Dense $(M'N', M'N')$))
8	LayerNormalization $(\mathfrak{y}_{\text{layer 3}} + \mathfrak{y}_{\text{layer 7}})$
9	Softmax (Dense $(M'N', M'N')$)

The ReLU function is defined as $f_{ReLU}(x) = \max\{0, x\}$, and Dense d_I, d_O represents a fully connected layer with d_I as the input data dimension and d_O as the output data dimension. $Dropout(\delta)$ denotes a dropout operation within the NN layer, applied with probability δ, and Softmax is defined for generating probability measure outputs. The outputs from the 3rd and 7th layers are combined at the 8th layer via a residual connection.

6.4 Usage Example and Results

This repository is structured in the following way:

- *Configs/config.yaml* contains the training and system configurations performing different experiments

- *model* contains all the NN models
- *utils* contains common utility functions
- *weights* contains weights of pre-trained neural receivers for different configuration files
- *results* contains pre-computed BLER performance results

The workflow consists of the following steps:

5. *DataProcessing.py* to adjust data processing and convert MATLAB datasets data into .npz format
6. Define simulation parameters in *config.yaml*: this includes the entire simulation parameters including training/evaluation parameters
7. *Train.py* generate dataset, data loader, split the data for training and validation
8. *Trainer.py* inherit torch.nn.Module to pass the data to the model for training
9. *DistanceDecoding.py* to test NN decoder

We can examine three wide beam codebook designs: the common codebook for wide beam codebook design (WB), wide beam codebook design incorporating a circular-shift operation (CSWB), and wide beam codebook design featuring partial random coding (PR-WB). For the decoding technique, we consider two methods: the proposed ML-based decoder and the non-ML method (DD). As illustrated in Figure 11 and Figure 12, the cumulative distribution function (CDF) of Ersrp is computed using the testing data for both fixed and optimal UE beam selection accordingly. Additionally, the y-axis value at Ersrp~=0 dB indicates the prediction accuracy. The results indicate that: (1) the ML decoder significantly outperforms DD across all wide beam codebook designs; (2) the proposed CSWB and PR-WB models surpass the WB for each decoder method; (3) In our configuration, CS-WB with DD and PR-WB with DD demonstrate comparable performance to WB with ML, highlighting the importance of the wide beam design for refined beam prediction.

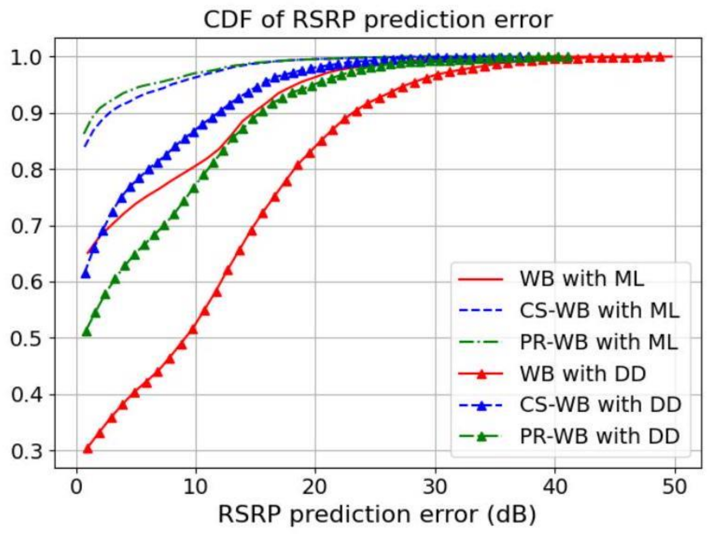

Figure 12: CDF of RSRP error for refined beam prediction with NN decoder with fixed UE receive beam selection

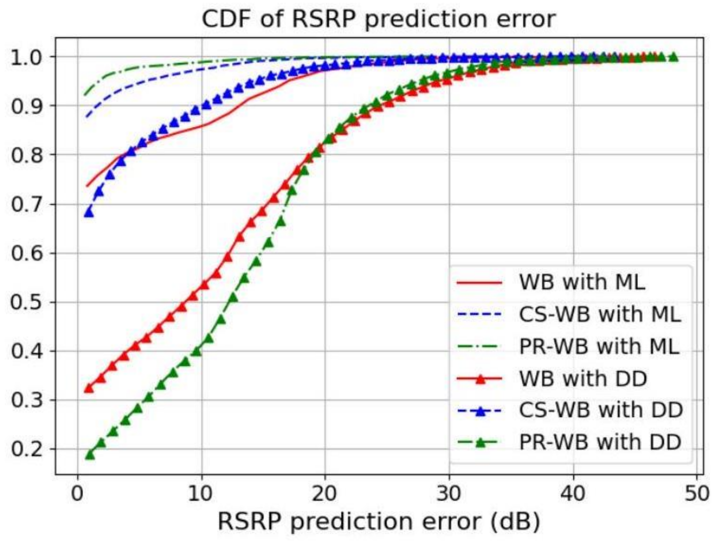

Figure 13: CDF of RSRP error for refined beam prediction with NN decoder with Optimal UE receive beam selection

6.5 Remarks

This repository is made available to allow reproducibility of the results and possible extension of the proposed AI algorithm for refined beam prediction.

7 References

[1] C. Sebastian, F. A. Aoudia, J. Hoydis, A. Oeldemann, A. Roessler, T. Mayer and A. Keller, “A Neural Receiver for 5G NR Multi-user MIMO,” in *IEEE Globecom Workshops*, 2023.

[2] R. Wiesmayr, S. Cammerer, F. A. Aoudia, J. Hoydis, J. Zakrzewki and A. Keller, “Design of a Standard-Compliant Real-Time Neural Receiver for 5G NR,” *arxiv preprint,* 2024.

[3] Q. Yang and S. J. Pan, “A Survey on Transfer Learning",” *IEEE Transactions on Knowledge & Data Engineering, ,* vol. 22, no. 10, pp. 1345-1359,, 2010.

[4] J. Hoydis, S. Cammerer, F. {Ait Aoudia}, A. Vem, N. Binder, G. Marcus and A. Keller, “Sionna: An Open-Source Library for Next-Generation Physical Layer Research,” ArXiv, 2022.

[5] C. Sung-E, R. Nancy and J. Tara, “Active Learning and CSI Acquisition for mmWave Initial Alignment,” *IEEE Journal on Selected Areas in Communications,* 2019.

[6] H. Haitham, A. Omid, R. Michael, A. Mohammed, K. Dine and I. Piotr, “Fast Millimeter Wave Beam Alignment,” in *ACM Special Interest Group on Data Communication*, 2018.

[7] S. Foad, J. Tao, C. Wei and Y. Wei, “Active Sensing for Communications by Learning,” *IEEE Journal on Selected Areas in Communications,* 2022.

[8] H. Yuqiang and G. A. Jeffrey, “Grid-Free MIMO Beam Alignment Through Site-Specific Deep Learning,” *IEEE Transactions on Wireless Communications,* 2024.

[9] Z. Andrea, B. Nicola, C. Angelo, V. Lorenzo and Z. Michele, “Internet of Things for Smart Cities,” *IEEE Internet of Things Journal,* 2014.

[10] S. R. Theodore, R. M. George, K. S. Mathew and S. Shu, “Wideband millimeter-wave propagation measurements and channel models for future wireless communication system design,” *IEEE Transactions on Communications,* 2015.

[11] G. Marco, P. Michele, R. Arnab, C. Douglas and Z. Michele, “A tutorial on beam management for 3GPP NR at mmWave Frequencies,” *IEEE Communications Survey and Tutorials,* 2019.

[12] S. Jiho, C. Junil and J. L. David, “Common codebook millimeter wave beam design: Design beams for both sounding and communication with uniform planar arrays,” *IEEE Transactions on Communications,* 2017.

[13] W. Cui and W. Yu, “Transfer Learning with Input Reconstruction Loss,,” in " *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Rio de Janeiro, 2022,.

[14] U. Uyoata and R. Adeogun, “Transfer Learning for a Neural Receiver,” AAU, August 2024. [Online]. Available: https://github.com/ramonadeog/CENTRIC-AAU/tree/main. [Accessed 21 August 2024].