Neural Networks Based Beam Codebooks: Learning mmWave Massive MIMO Beams that Adapt to Deployment and Hardware

Muhammad Alrabeiah, Yu Zhang, and Ahmed Alkhateeb

***Abstract*—Millimeter wave (mmWave) and massive MIMO systems are intrinsic components of 5G and beyond. These systems rely on using beamforming codebooks for both initial access and data transmission. Current beam codebooks, however, generally consist of a large number of narrow beams that scan all possible directions, leading to large training overhead. Further, these codebooks do not normally account for hardware impairments or possible non-uniform array geometries, and their calibration process is expensive. To overcome these limitations, this paper develops an efficient online machine learning framework that learns how to adapt the codebook beam patterns to the specific deployment, surrounding environment, user distribution, and hardware characteristics. This is done by designing a novel *complex-valued neural network* architecture in which the neuron weights directly model the beamforming weights of the analog phase shifters, accounting for the key hardware constraints. This model learns the codebook beams through online and self-supervised training avoiding the need for explicit channel state information. This respects the practical situations where the channel is imperfect or hard to obtain. Simulation results highlight the capability of the proposed solution in learning environment and hardware aware beam codebooks, which reduce the training overhead and improve the robustness against possible hardware impairments.**

***Index Terms*—Beam codebook learning, online learning, self-supervised, neural networks, beyond 5G, millimeter wave, massive MIMO.**

I. Introduction

Millimeter wave (mmWave) MIMO is an essential ingredient of the future wireless communication networks—from 5G to IEEE 802.11ay and beyond [2]–[4]. These systems use large antenna arrays to obtain enough beamforming gains and guarantee sufficient receive signal power. Due to the cost and power consumption of the mixed-signal circuits at the high frequency bands, though, fully-digital transceiver architectures that assign an RF chain per antenna are not feasible [5]. Instead, these mmWave systems resort to analog-only or hybrid analog/digital architectures [6] to implement the beamforming/combining functions. Further, because of the hardware constraints on these large-scale MIMO systems and the difficulty in channel estimation and feedback, they typically adopt pre-defined single-lobe beamforming codebooks (such as DFT codebooks [7]) that scan all possible directions for both initial access and data transmission. Examples of using these codebooks include the Synchronization Signal Block (SSB) beam sets and Channel State Information Reference Signal (CSI-RS) codebooks in 5G [4], and hierarchical beam patterns in IEEE 802.11ad [8]. The classical beam-steering codebooks, however, have several drawbacks: (i) They incur high beam training overhead by scanning all possible directions even though many of these directions may never be used, (ii) they normally have single-lobe beams which may not be optimal, especially in non-line-of-sight (NLOS) scenarios, and (iii) they are typically predefined and do not account for possible hardware imperfections (such as phase mismatch or arbitrary array geometries) with very expensive calibration processes [9]. **To overcome these limitations, we propose a novel online machine learning framework that learns how to adapt the codebook beam patterns to the surrounding environment, the user distribution, and the given hardware of the specific base station deployment**—building what we call environment and hardware aware beam codebooks.

Muhammad Alrabeiah, Yu Zhang and Ahmed Alkhateeb are with Arizona State University (Email: malrabei, y.zhang, alkhateeb@asu.edu). This work is supported by the National Science Foundation under Grant No. 1923676. A conference version of this paper has been published in [1].

A. Prior Work:

Designing MIMO beamforming codebooks has been an important research and development topic for a long time at both academia and industry [10]–[17]. The motivation for all this prior work has been mainly to enable efficient limited-feedback operation in MIMO systems. For example, the authors of [10], [11] investigated the design of beamforming codebooks for MISO communication systems with Rayleigh channels. The same problem was then considered in [12], [13] for spatially and temporally correlated channels. For systems with multiple antennas at both the transmitters and receivers, [14], [15] developed precoding/combining codebooks and analyzed the system performance under various channel models. The use of beam codebooks have been also adopted by several cellular and wireless LAN standards [16], [17]. The codebook approaches in [10]–[17], however, were generally designed to optimize the feedback of small-scale MIMO and are hard to extend to massive MIMO systems without the requirement of huge codebook sizes and large training overhead. Further, the codebooks in [10]–[17] adopted fully-digital architectures and did not consider the hardware constraints at the transmitter/receiver arrays which could highly affect the design of these codebooks. Incorporating these constraints is essential for the development of efficient mmWave MIMO codebooks.

For mmWave systems, [6], [7], [18], [19] developed a set of new beamforming codebooks for analog-only and

hybrid analog/digital architectures. Besides, as beam alignment and tracking play a pivotal role in mmWave systems [20], [21], in [18], narrow-beam codebooks were developed to aid the beam training processing mmWave systems. The narrow beams, however, may lead to large training overhead. This motivated designing hierarchical codebooks in [6], [7], [22] that consist of different levels of beam widths. For frequency selective channels, [19] developed iterative hybrid analog/digital beamforming codebooks. The codebooks in [6], [7], [18], [19], however, have several limitations. First, they were generally designed for unconstrained architectures and then approximated for these constraints, i.e., they were not particularly optimized for these hardware constraints. Second, they were mainly designed to have single-lobe narrow beams that cover all the angular directions and are not adaptive to the particular deployment characteristics (surrounding environment, user distributions, etc.), which requires large training overhead. Further, these codebooks assumed fully-calibrated uniform arrays and experience high distortion in practical hardware with fabrication impairments. All that motivated the development of environment and hardware aware codebook learning strategies, which is the focus of this paper.

B. Contribution:

In this paper, we consider hardware-constrained large-scale MIMO systems and propose an artificial neural network based framework for learning environment and hardware adaptable beamforming codebooks. The main contributions of the paper can be summarized as follows

- First, we design a supervised machine learning model that can learn how to adapt the patterns of the codebook beams based on the surrounding environment and user distribution. This is done by developing a novel *complex-valued neural network* architecture in which the weights directly model the beamforming/combining weights of the analog phase shifters. The proposed model accounts for the key hardware constraints such as the phase-only, constant-modulus, and quantized-angle constraints [5]. The training process was designed to approach the performance of equal-gain transmission/combining [23], which is the upper bound of the analog-only beamforming solutions.
- Then, we develop a second neural network architecture that relies on online and self-supervised learning and avoids the need for explicit channel state information. This respects the practical situations where the channel state information is either unavailable, imperfect, or hard to obtain, especially in the presence of hardware impairments. The developed architecture learns in an online and self-supervised fashion how to adapt the codebook beam patterns to suit the surrounding environment, user distribution, hardware impairments, and unknown antenna array geometry.
- Finally, we extensively and comprehensively evaluate the performance of the proposed codebook learning approaches based on the publicly-available DeepMIMO dataset [24]. These experiments adopt both outdoor and indoor wireless communication scenarios at different signal-to-noise ratios (SNRs) and codebook sizes. The evaluation is done for both uniform-perfect arrays and arrays with arbitrary geometries and hardware impairments. It focuses on demonstrating the performance of the proposed solutions (supervised and self-supervised), empirically studying the convergence of those solutions, and benchmarking them to a classical codebook design approach that requires a form of channel estimation.

The simulation results verify the effectiveness of the proposed solutions in providing the sought-after environment and hardware awareness. In particular, the proposed solutions show significant improvements compared to classical beam-steering codebooks in several cases: (i) For arbitrary user distributions in which our approaches learn how to adapt the beams to focus on where the users are and significantly reduce the required beam training overhead, (ii) for NLOS scenarios with multiple equally-strong paths where the developed codebook learning solutions learn multi-lobe beams that achieve much higher data rates, and (iii) for arrays with hardware impairments or unknown geometries, where our neural networks learn how to adapt the beam patterns for the given arrays and mitigate the impact of hardware impairments. All that highlights a promising direction where machine learning can be integrated into the communication systems to develop deployment and hardware specific beam codebooks. Moreover, it is worth pointing out that the developed solutions could be leveraged for the analog precoding part of the hybrid architectures. However, by better leveraging the existence of more than one RF chain in the hybrid architectures, there might be more advanced beam codebook design solutions, which is an interesting topic for future extensions.

II. System and Channel Models

In this section, we describe in detail our adopted system and channel model. Further, we describe how the model considers arbitrary arrays with possible hardware impairments.

A. System Model

We consider the communication system shown in Fig. 1 where a base station (BS) with M antennas is deployed in a certain environment and is capable of serving both the LOS and NLOS mobile users in this environment. For simplicity, we assume that the users have single antennas. The proposed solutions in this paper, however, could be extended to the case with multi-antenna users. Next, considering the uplink transmission, if the user transmits a symbol $s \in \mathbb{C}$, then the received signal at the BS after combining can be expressed as

$$y = \mathbf{w}^H \mathbf{h} s + \mathbf{w}^H \mathbf{n}, \tag{1}$$

where the transmitted symbol satisfies the average power constraint $\mathbb{E}\left[|s|^2\right] = P_s$ and $\mathbf{n} \sim \mathcal{N}_{\mathbb{C}}\left(0, \sigma_n^2 \mathbf{I}\right)$ is the receive noise vector at the BS. The $M \times 1$ vector $\mathbf{h} \in \mathbb{C}^{M \times 1}$ denotes the uplink channel between the mobile user and the BS antennas and $\mathbf{w}$ represents the BS combining vector. Given the cost and power consumption of the mixed-signal components at the mmWave frequencies, it is hard to dedicate

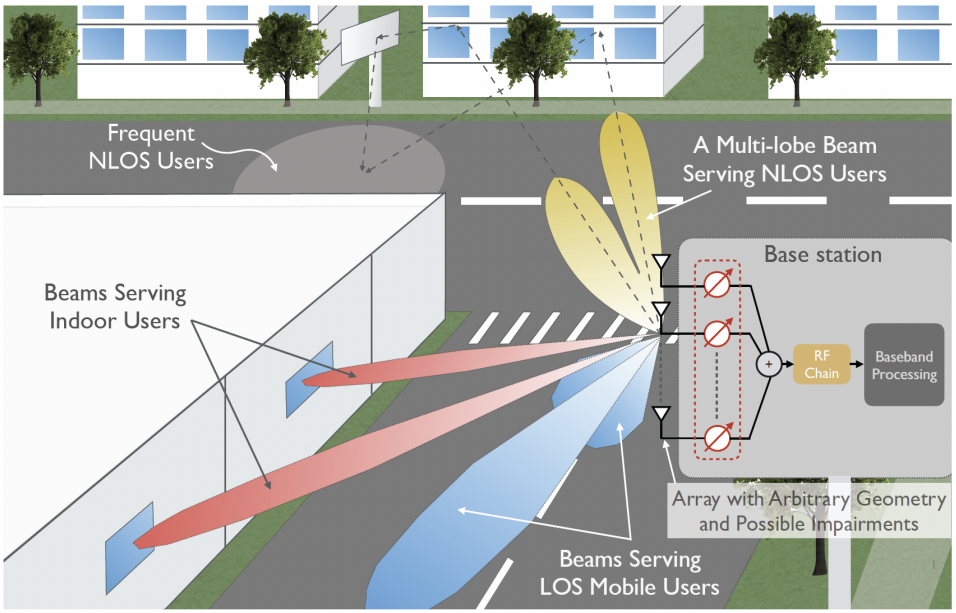


Fig. 1. The adopted system model where a base station of M antennas can communicate with LOS or NLOS users using a beam codebook. The proposed machine learning model in this paper learns how to efficiently adapt the codebook beams based on the given deployment, user distributions, and hardware characteristics.

an RF chain for each antennas and apply fully-digital precoding/combining at mmWave massive MIMO systems. Alternatively, mmWave base stations adopt analog-only or hybrid analog-digital beamforming approaches that move all or some of the beamforming/combining processing to the RF domain [5]. To account for that, we assume that the BS employs an analog-only architecture where the beamforming/combining is implemented using a network of phase shifters as depicted in Fig. 1. With this architecture, the combining vector $\mathbf{w}$ can be written as

$$\mathbf{w} = \frac{1}{\sqrt{M}} \left[e^{j\theta_1}, e^{j\theta_2}, \ldots, e^{j\theta_M}\right]^T, \tag{2}$$

which can only perform phase shift to the signal received by each antenna.

B. Channel Model

We adopt a general geometric channel model for $\mathbf{h}$ [25], [26]. Assume that the signal propagation between the mobile user and the BS consists of L paths. Each path ℓ has a complex gain α_ℓ (that includes the path-loss) and an angle of arrival ϕ_ℓ. Then, the channel can be written as

$$\mathbf{h} = \sum_{\ell=1}^{L} \alpha_\ell \mathbf{a}(\phi_\ell), \tag{3}$$

where $\mathbf{a}(\phi_\ell)$ is the array response vector of the BS. The definition of $\mathbf{a}(\phi_\ell)$ depends on the array geometry and hardware impairments, which will be discuss in the next subsection. Without loss of generality, the first path in (3), i.e., $\ell = 1$, is always assumed to represent a direct path from the user to the base station. Hence, to model LOS and NLOS channels in this paper, (3) is modified to incorporate a binary vector $\mathbf{b} = [b_1, \ldots, b_L]$ as follows

$$\mathbf{h} = \sum_{\ell=1}^{L} b_\ell \left[\alpha_\ell \mathbf{a}(\phi_\ell)\right], \tag{4}$$

where $b_\ell \in \{0, 1\}$, $\forall \ell \in \{1, \ldots, L\}$. According to (4), $\mathbf{h}$ is considered LOS when $b_1 = 1$ and NLOS otherwise.

C. Arbitrary Array Geometry and Hardware Impairments

Most of the prior work on mmWave signal processing has assumed uniform antenna arrays with perfect calibration and ideal hardware [5]–[7], [18]. In this paper, we consider a more general antenna array model that accounts for arbitrary geometry and hardware imperfections. We show that our online beam codebook learning approaches can efficiently learn beam patterns for these arrays and adapt to their particular characteristics. This leads to several advantages for these systems since (i) there are scenarios where designing arbitrary arrays is needed, for example, to improve the angular resolution or enhance the direction-of-arrival estimation performance [27], [28], (ii) the fabrication process of large mmWave arrays normally has some imperfections, and (iii) because the calibration process of the mmWave phased arrays is an expensive process that requires special high-performance RF circuits [9]. While the codebook learning solutions that we develop in this paper are general for various kinds of arrays and hardware impairments, we evaluate them in Section VIII with respect to two main characteristics of interest, namely non-uniform spacing and phase mismatch between the antenna

elements. For linear arrays, the array response vector can be modeled to capture these characteristics as follows

$$\mathbf{a}(\phi_\ell) = \Big[e^{j(kd_1\cos(\phi_\ell)+\Delta\theta_1)}, e^{j(kd_2\cos(\phi_\ell)+\Delta\theta_2)}, \ldots, e^{j(kd_M\cos(\phi_\ell)+\Delta\theta_M)}\Big]^T, \quad (5)$$

where k is the wave number, d_m is the position of the m-th antenna, and $\Delta\theta_m$ is the additional phase shift incurred at the m-th antenna (to model the phase mismatch). Without loss of generality, we assume that d_m and $\Delta\theta_m$ are **fixed yet unknown random realizations**, obtained from a truncated Gaussian distribution denoted as $\tilde{\mathcal{N}}\left((m-1)d, \sigma_d^2; (m-\frac{3}{2})d, (m-\frac{1}{2})d\right)$ and a Gaussian distribution $\mathcal{N}\left(0, \sigma_p^2\right)$, respectively. The probability density function (PDF) of the truncated Gaussian distribution $\tilde{\mathcal{N}}$ is given by

$$f(x) = \frac{1}{\sigma_d} \frac{\phi\left(\frac{x-(m-1)d}{\sigma_d}\right)}{\Phi\left(\frac{d}{2\sigma_d}\right) - \Phi\left(-\frac{d}{2\sigma_d}\right)}, \quad (6)$$

where $(m-\frac{3}{2})d \leq x \leq (m-\frac{1}{2})d$. $\phi(\xi) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\xi^2}$ is the PDF of the standard Gaussian distribution and $\Phi(\cdot)$ is its cumulative distribution function.

III. Problem Definition

In this paper, we investigate the design of mmWave beamforming codebooks that are adaptive to the specific deployment (surrounding environment, user distribution/traffic, etc.) and the given hardware (array geometry, hardware imperfections, etc.), as shown in Fig. 1. Next, we formulate the beam codebook optimization problem before showing in Sections IV-V how neural network based machine learning can provide efficient approaches for learning adaptive codebooks. Given the system and channel models described in Section II, the SNR after combining for user u can be written as

$$\mathsf{SNR}_u = \frac{\left|\mathbf{w}^H\mathbf{h}\right|^2}{\|\mathbf{w}\|_2^2}\rho, \quad (7)$$

with $\rho = \frac{P_s}{\sigma^2}$. If the combining vector $\mathbf{w}$ is selected from a codebook $\boldsymbol{\mathcal{W}}$, with cardinality $|\boldsymbol{\mathcal{W}}| = N$, then, the maximum achievable SNR for use u is obtained by the exhaustive search over the beam codebook as

$$\mathsf{SNR}_u^\star = \rho \max_{\mathbf{w}\in\boldsymbol{\mathcal{W}}} \left|\mathbf{w}^H\mathbf{h}\right|^2, \quad (8)$$

where we set $\|\mathbf{w}\|_2^2 = 1$ as these combining weights are implemented using only phase shifters with constant magnitudes of $1/\sqrt{M}$, as described in (2). Our objective in this paper is to design the codebook $\boldsymbol{\mathcal{W}}$ to maximize the SNR averaged over the candidate set of user channels $\boldsymbol{\mathcal{H}}$, which are the channels of the candidate users in the environment surrounding the deployed BS. This problem can then be written as

$$\boldsymbol{\mathcal{W}}_{\mathsf{opt}} = \arg\max_{\boldsymbol{\mathcal{W}}} \sum_{\mathbf{h}\in\boldsymbol{\mathcal{H}}} \left(\max_{\substack{\mathbf{w}_n\in\boldsymbol{\mathcal{W}},\\ n=1,\ldots,N}} \left|\mathbf{w}_n^H\mathbf{h}\right|^2 \right), \quad (9)$$

$$\text{s.t.} \left|[\mathbf{w}_n]_m\right| = \frac{1}{\sqrt{M}}, \forall m = 1,\ldots,M, n = 1,\ldots,N, \quad (10)$$

where the constraint in (10) is imposed to uphold the phase-shifters constraint, i.e., the analog beamformer can only perform phase shifts to the received signal but is not capable of adapting the gain. It is worth mentioning here that while we are focusing on receive beamforming design in this paper, the same solution can be used for transmit codebook design by acquiring SNR feedback from the users. Moreover, in order to obtain the up-to-date codebooks, in practice, the channel dataset is expected to be refined and updated over time to match the large time-scale changes in the environment and use traffic statistics.

The objective of problem (9) is to find the beam codebook that maximizes the average SNR gain for all the candidate users. Since we only have a finite beamforming codebook, which is far less than the number of users, it is impossible to achieve the maximum combining SNR for each user (which is given by the equal-gain combining [23]). In this sense, we might expect to find a codebook such that each beamformer serves a group of users that share similar channels. Due to the large number of channels in $\boldsymbol{\mathcal{H}}$ as well as the non-convex constraints (10), problem (9) in general is very hard to solve by using the classical optimization methods and beamforming design approaches [6], [19], [29], [30]. Therefore, and motivated by the powerful learning and optimization capabilities of neural networks, we consider leveraging neural network based machine learning to efficiently solve the optimization problem (9). Depending on whether the channel state information is available or not, two different machine learning frameworks are designed, namely supervised and self-supervised solutions, in Sections IV and V to learn beam codebooks that adapt to the given deployment and hardware—generating what we call environment and hardware aware beam codebooks.

IV. Supervised Machine-Learning Solution

Designing environment-aware mmWave beam codebooks requires an adaptive and data-driven process. Data collected from the environment surrounding a base station, like channels and/or user-received power, is a powerful source of information as it encodes information on user distributions and users' multi-path signatures. Such data could be used to tailor the beamforming codebook to those users and that environment. The challenge here is the need for a system capable of sifting through the data, analyzing it, and designing the codebook in a manner that respects the phase-shifter constraint. This clearly calls for a system with a sense of intelligence.

This section addresses that challenge and proposes an elegant solution that is environmentally adaptable, data-driven, and hardware compatible. In its core, this solution relies on machine learning and, in particular, artificial neural networks [31]. It follows a supervised learning approach to analyze the channel structure and learn the phases of the *suitable* beamforming vectors. Its elegance stems from the way it learns the codebook; the weights of the neural network directly relate to the angles of the phased arrays, making them the actual parameters of the network. Therefore, during every training cycle (forward and backward passes) the codebook will be updated directly.

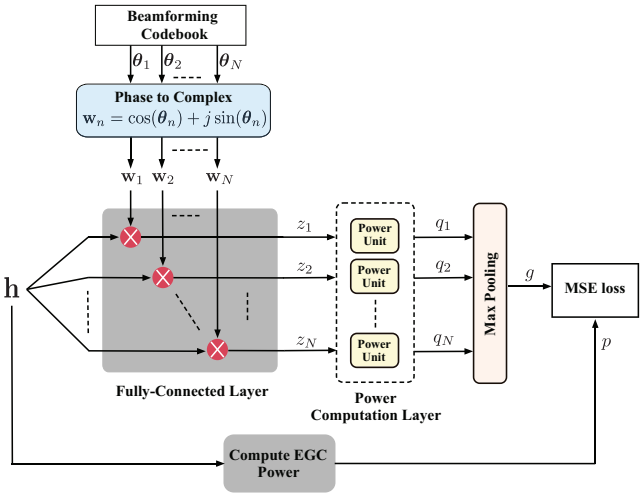


Fig. 2. This schematic shows the overall architecture of the neural network used to learn the beamforming codebooks. It highlights the network architecture and the auxiliary components, equal-gain-combining and MSE-loss units, used during the training process. It also gives a slightly deeper dive into the inner-workings of the cornerstone of this architecture, the complex-valued fully-connected layer.

A. Model Architecture

Before going into the details of how a codebook is learned, it is important to explain the architecture of the proposed neural network and its relation to the optimization problem in (9). This architecture consists of three main components, as depicted in Fig. 2. Those components are the complex-valued fully-connected layer, the power-computation layer, and finally the max-pooling layer. A forward pass through these three layer is equivalent to evaluating the cost function of (9) over a single channel $\mathbf{h}$.

1) Complex-valued fully-connected layer: The first layer consists of N neurons that are capable of performing complex-valued multiplications and summations. Each neuron, as shown in Fig. 2, learns one beamforming vector and performs inner product with the input channel vector. Formally, this is described by the following matrix multiplication

$$\mathbf{z} = \mathbf{W}^H \mathbf{h}, \tag{11}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N] \in \mathbb{C}^{M \times N}$ is the beamforming codebook, $\mathbf{h}$ is a user's channel vector, and $\mathbf{z} \in \mathbb{C}^{N \times 1}$ is the vector of the combined received signal. This equation could be re-written in the following block matrix form

$$\begin{bmatrix} \mathbf{z}^{\mathrm{r}} \\ \mathbf{z}^{\mathrm{im}} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^{\mathrm{r}} & -\mathbf{W}^{\mathrm{im}} \\ \mathbf{W}^{\mathrm{im}} & \mathbf{W}^{\mathrm{r}} \end{bmatrix}^T \begin{bmatrix} \mathbf{h}^{\mathrm{r}} \\ \mathbf{h}^{\mathrm{im}} \end{bmatrix}, \tag{12}$$

where $\mathbf{z}^{\mathrm{r}}, \mathbf{z}^{\mathrm{im}} \in \mathbb{R}^{N \times 1}$ are the real and imaginary parts of $\mathbf{z}$, $\mathbf{W}^{\mathrm{r}}, \mathbf{W}^{\mathrm{im}} \in \mathbb{R}^{M \times N}$ are matrices containing the real and imaginary components of the elements of $\mathbf{W}$, and finally, $\mathbf{h}^{\mathrm{r}}, \mathbf{h}^{\mathrm{im}} \in \mathbb{R}^{M \times 1}$ are the real and imaginary components of the channel vector $\mathbf{h}$. What is interesting about (12) is that it provides a peek behind the curtains to the inner-workings of the complex-valued fully-connected layer.

Contrary to the norm in designing neural networks, the elements of the beamforming matrix $\mathbf{W}$ are not the weights of the fully-connected layer. Instead, they are derived from the actual neural network weights, which are the phase shifts making up the beamforming codebook. This is done through an embedded layer of phase-to-complex operations, as shown in Fig. 2. This layer transforms the phase shifts into unit-magnitude complex vectors by applying elements-wise $\cos$ and $\sin$ operations and scale them by $1/\sqrt{M}$ as follows

$$\mathbf{W} = \frac{1}{\sqrt{M}} \left(\cos\left(\mathbf{\Theta}\right) + j * \sin\left(\mathbf{\Theta}\right) \right), \tag{13}$$

where $\mathbf{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N]$ is an $M \times N$ matrix of phase shifts, and $\boldsymbol{\theta}_n = [\theta_{1n}, \ldots, \theta_{Mn}]^T, \forall n \in \{1, \ldots, N\}$ is a single phase vector. The use of this embedded layer is the network's way of learning beamforming vectors that **respect the phase shifter constraint**.

2) Power-computation layer: The output of the complex-valued fully-connected layer is fed into the power-computation layer. It performs element-wise absolute square operation and outputs a real-valued vector $\mathbf{q}$ given by

$$\mathbf{q} = [q_1, q_2, \ldots, q_N]^T = \left[|z_1|^2, |z_2|^2, \ldots, |z_N|^2\right]^T, \tag{14}$$

which has the received power of each beamformer in the codebook.

3) Max-pooling layer: The power of the best beamformer is, finally, found by the last layer, the max-pooling layer. It performs the following element-wise $\max$ operation over the elements of $\mathbf{q}$

$$g = \max\left\{q_1, q_2, \ldots, q_N\right\}, \tag{15}$$

and outputs g, which is the power of the best beamformer. This value is used to assess the quality of the codebook by comparing it to a desired receive power value. The details on what this desired value is and how the quality is assessed are detailed in the following subsection.

B. Learning Codebooks

With the neural network architecture in mind, it is time to delve into the details of how a codebook is learned. This first proposed solution, as its name states, follows a supervised learning approach. In such approach, a machine learning model is trained using pairs of inputs and their desired responses, which constitute the training dataset.

1) Desired response: For the beamforming problem in hand, the inputs to the model are the users' channels as they are the communication quantity that drives the beamforming design process. As training targets, there are many possible desired responses that could be used, and the choice between them should be made based on what the models needs to learn. In this paper, equal gain combining (EGC) is adopted as the desired response. This choice is based on the fact that equal gain combining respects the phase shifters constraint. It is the beamforming that achieves optimal SNR performance when there are no restrictions on the codebook size. Further, equal gain combining constitutes an upper bound for the received power of fully-analog transceivers. The equal-gain combining beamformer is obtained using the phase component of every user's channel as follows

$$\mathbf{w}_{\mathrm{EGC}} = \frac{1}{\sqrt{M}} \left[e^{j\angle h_1}, e^{j\angle h_2}, \ldots, e^{j\angle h_M}\right]^T, \tag{16}$$

where $\angle$ stands for the phase of a complex number. Using equal gain combining beamformers, the desired response for each user can be computed as follows

$$p = \left|\mathbf{w}_{\mathrm{EGC}}^{H}\mathbf{h}\right|^{2} = \frac{1}{M}\left\|\mathbf{h}\right\|_{1}^{2}, \tag{17}$$

where $\|\cdot\|_1$ is the $L1$ norm. Putting the users' channels and their equal-gain combining gains together provides the training dataset $\mathcal{S}_t$.

2) Model background training: Using the set $\mathcal{S}_t$, the model is trained in the background by undergoing multiple forward-backward cycles. In each cycle, a *mini-batch* of complex channel vectors and their equal-gain combining responses is sampled from the training set. The channels are fed sequentially to the model and a forward pass is performed as describe in Section IV-A. For each channel vector in the batch, the model combines it with the currently available N beamforming vectors and outputs the power of the best beamformer for that channel. The quality of the best combiner is assessed by measuring how close its beamforming gain to that of the channel equal-gain combiner, obtained by (17). A Mean-Squared Error (MSE) loss is used as a metric to assess the quality of the codebook over the current mini-batch. Formally,

$$\mathcal{L} = \frac{1}{B}\sum_{b=1}^{B}(g_b - p_b)^2, \tag{18}$$

where g_b is the output of the max-pooling layer for the b-th data pair in the mini-batch, and B is the mini-batch size. The error signal (derivative of the loss (18) with respect to each phase vector $\boldsymbol{\theta}_n \in \boldsymbol{\Theta}$) is propagated back through the model to adjust the phases of the combining vectors [32] [33], making up what is usually referred to as the backward pass or *backpropagation*. This is formally expressed by the chain rule of differentiation:

$$\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_n}\right)^{T} = \frac{\partial \mathcal{L}}{\partial g}\cdot\left(\frac{\partial g}{\partial \mathbf{q}}\right)^{T}\cdot\frac{\partial \mathbf{q}}{\partial \mathbf{z}}\cdot\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}. \tag{19}$$

In mathematical terms, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_n}$ does not exist, for the factor $\frac{\partial \mathbf{q}}{\partial \mathbf{z}}$ does not satisfy the Cauchy-Riemann equations [34], meaning that $\mathbf{q}$ as a function of the complex vector $\mathbf{z}$ is not complex differentiable (*holomorphic*). However, the issue could be resolved to enable backpropagation. The details of that and how the derivatives are computed are discussed in Appendices A and B. Computing the partial derivative of the loss with respect to phase vector $\boldsymbol{\theta}_n$ allows the backward pass to modify the codebook $\boldsymbol{\Theta}$ and make it adaptive to the environment. The update equation generally depends on the solver used to carry out the training, e.g., Stochastic Gradient Descent (SGD) and ADAptive Moment estimation (ADAM) to name two, but in its simplest form, it could be given by

$$\boldsymbol{\theta}_{n\mathsf{new}} = \boldsymbol{\theta}_{n\mathsf{cur}} - \eta\cdot\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_n} \tag{20}$$

where η is the optimization step size, commonly known as the learning rate in machine learning, and $\boldsymbol{\theta}_{n\mathsf{new}}$ and $\boldsymbol{\theta}_{n\mathsf{cur}}$ are, respectively, the new and current n-th phase vector of the codebook.

C. Learning Quantized Codebook

Restriction on the resolution of the phase shifter is common in many mmWave implementations. This imposes limits on the number of phase vectors that could be realized by a system, giving rise to learning quantized codebooks. The proposed solution is actually capable of learning such codebooks. This could be achieved using a quantize-while-training approach, similar to that in [35] [36]. The training process, presented as forward and backward passes in Sections IV-A and IV-B, respectively, is tweaked to incorporate a k-means quantizer. The quantizer is implemented right after updating the parameters of the network in (20). It takes the phase-vector codebook and vectorize it $\tilde{\boldsymbol{\Theta}} = \left[\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T\right]_{1\times NM}$, and then, it applies k-means on the elements of $\tilde{\boldsymbol{\Theta}}$. The returned cluster centroids, which are a set of scalars, define the new finite set of angles the phase shifters need to realize. The size of that set (number of centroids) is determined by the phase shifter resolution, Q bits.

Remark: the proposed quantizer is transparent to the backward operation described in Section IV-B. This means it does not change the backpropagation equation in (19), and the quantized values of the phases are directly substituted into the partials $\partial \mathbf{z}/\partial \boldsymbol{\theta}_n, \forall n \in \{1, \ldots, N\}$.

V. Self-Supervised Machine Learning Solution

In this section, an alternative neural network architecture is proposed to perform the same codebook learning process without requiring accurate channel knowledge. The motivations for developing this model are two fold: (i) The existence of hardware impairments prevents accurate channel acquisition in mmWave systems as obtaining them could be a difficult process that requires very large training overhead [6], and (ii) the need for channel information implies that the codebook learning process has to be performed *offline*, which may not be favorable for swift adaptability. To address these problems, we propose a novel *self-supervised* learning solution. This solution, as the name suggests, works in a self-sufficient fashion instead of requiring the supply of a desired response for every training channel.

A. Self-supervision via Clustering

Before diving into the details of the new proposed architecture, it is helpful to first illustrate the basic idea of this design. The motivations for this new model are rooted in the lack of desired responses and the need for online learning. As a result, the model should only rely on itself to learn how to adjust the codebook beams such that the performance is improved. This is accomplished by tapping into an intrinsic feature the final codebook must have, channel space partitioning; as explained in Section III, the codebook has fixed size, and, therefore, each beamformer is ultimately expected to be optimized to serve a set of users in the environment. This is mathematically equivalent to partitioning $\boldsymbol{\mathcal{H}}$ into subsets of channels

$$\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}_1 \cup \boldsymbol{\mathcal{H}}_2 \cup \cdots \cup \boldsymbol{\mathcal{H}}_N, \tag{21}$$

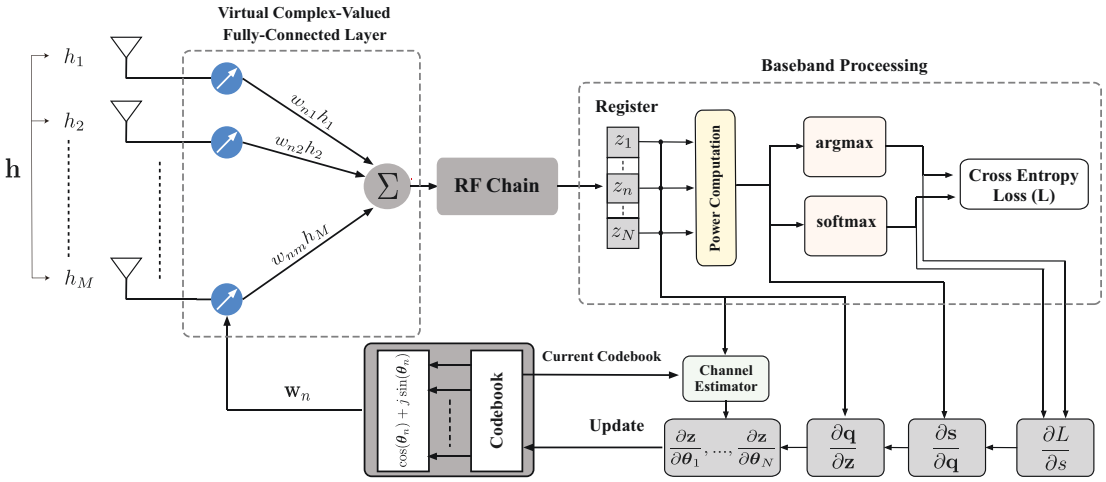


Fig. 3. The proposed self-supervised framework as it is envisioned in practice. The solution is integrated into the different components of a mmWave base station with analog architecture. The phase shifters with the combiner all together form a fully-connected layer, and the rest of the layers are implemented into the base-band processing unit.

where

$$\boldsymbol{\mathcal{H}}_{n'} \cap \boldsymbol{\mathcal{H}}_n = \emptyset, \ \forall n' \neq n \text{ and } n', n \in \{1, \ldots, N\}. \quad (22)$$

From a machine learning perspective, this partitioning could be translated into channel *clustering* where each beamformer is a cluster representative. Under this new view of the problem, the machine learning model generates its labels using the following strategy: For the received signal of an uplink pilot, it identifies the best beamforming vector in the current codebook, say $\mathbf{w}_n$ where $n \in \{1, \ldots, N\}$. Then, it adjusts the direction of that beamformer such that it results in higher beamforming gain with the current channel. Therefore, when a similar channel form the same partition, say $\boldsymbol{\mathcal{H}}_n$, is experienced, $\mathbf{w}_n$ is expected to be the best beamformer again, increasing its chance to be the representative for $\boldsymbol{\mathcal{H}}_n$. The technical details on how this is done are presented in the following couple of subsections, in which the model components, forward pass, and backward pass are explored.

B. Model Architecture

Fig. 3 presents a schematic of the proposed architecture as it is envisioned in a mmWave communication system. The following details a forward pass through the different components of this architecture:

1) Complex-Valued Fully-Connected Layer: similar to its supervised counterpart, the self-supervised network also adopts complex-valued fully-connected layer as its first layer. However, as integration into the communication system is in the core of this solution, the layer is implemented using the phase shifters, not a digital processor. As a result, it is referred to in Fig. 3 as a *virtual complex-valued fully-connected layer* (virtual layer for short). The function this layer implements is the same as that in Section IV-A1, and as such, its output is also given by (11). The main difference between this layer and that in Section IV-A1 comes in the implementation of the matrix vector multiplication. The virtual layer performs it by requiring the user to send a sequence of pilots, each of which is received with a different beamformer.

2) Register: the register buffers the received single of each beamformer until a full sweep across the codebook is completed. This temporary storage is essential as the following layers need to operate on the outputs of the virtual layers jointly.

3) Power-computation layer: once the system collects the whole outputs (all the beams in the codebook have been tried), those values are fed into the power-computation layer which calculates the beamforming gain for each beamformer using (14).

4) Softmax and argmax: this layer is where the self-supervised solution really differs from the supervised one. Instead of having a max-pooling layer, the output of power-computation layer is fed into two different layers, a softmax and an argmax. The former is employed to convert each beamforming gain to a "probability", which indicates how likely a beamformer is the optimal one to receive the user's signal given the current channel. Formally, having (14) as input, the n-th element of the output probability vector of the softmax layer $\mathbf{s} = [s_1, \ldots, s_N]^T$ can be expressed as

$$s_n = \frac{e^{|z_n|^2}}{\sum_{n=1}^{N} e^{|z_n|^2}}. \quad (23)$$

The argmax layer, on the other hand, outputs a one-hot vector $\mathbf{c} \in \{0, 1\}^N$, of the same dimension as $\mathbf{s}$, with 1 at the position where $\mathbf{s}$ attains its maximum value and with 0 at all other positions. This one-hot vector $\mathbf{c}$ is the self-generated label. It declares the best beamforming vector the representative of the cluster, and along with the output of softmax, they help tweak this best beamformer to make sure it has higher beamforming gain than other beamformers when it receives a similar channel in the future. This is accomplished by implementing a cross-entropy loss function. The following subsection will elaborate more on that loss and its role.

C. Learning Codebooks

After a forward pass, the model must do backpropagation to improve its performance, i.e., learning better beamforming

vectors. With the self-generated label and the probability vector, it is a matter of using that label to increase the probability of the currently selected beamformer.

1) Loss function: The first step to do backpropagation is to define a loss function that captures the objective of the model. As stated above, the model aims at clustering the channels and having the beamforming vectors in the codebook as representatives of those clusters. This is attained by a cross-entropy loss function given as

$$\mathcal{L}(\mathbf{s}, \mathbf{c}) = -\sum_{n=1}^{N} c_n \log s_n, \tag{24}$$

where $\mathbf{s}$ is the output of the softmax layer and $\mathbf{c}$ is the one-hot vector generated by argmax layer. This loss function makes the one-hot vector a target probability distribution for the model, and hence, it is not adjustable; the value of $\mathcal{L}$ must only be minimized by pushing the softmax distribution $\mathbf{s}$ to be as close to $\mathbf{c}$ as possible.

2) Backpropagation: The error signal is generated by differentiating the loss (24) with respect to each phase vector $\boldsymbol{\theta_n} \in \boldsymbol{\Theta}$. This error is backpropagated through the network to adjust the phases of all the beamforming vectors [32] [33] using the chain rule as follows

$$\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_n}\right)^T = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{s}}\right)^T \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{q}} \cdot \frac{\partial \mathbf{q}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}, \tag{25}$$

and (20) is used to update the phase vectors of the codebook. The implementation of this chain of derivatives is illustrated in Fig. 3. There are two issues with the error signal in (25). The first is similar to that issue encountered with the supervised model; $\mathbf{q}$ as a function of $\mathbf{z}$ is not *complex differentiable* or *holomorphic*, which implies that $\frac{\partial \mathbf{q}}{\partial \mathbf{z}}$ is not defined. The same argument developed for (19) and presented in Appendices A and B will be used here to obtain that partial. The second issue comes from the partial $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}$. Referring to (11), it is clear that computing $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}$ requires channel information, which is not explicitly available in this case. This is sidestepped with the help of a simple channel estimator described in the following subsection.

3) Channel Estimator: In order to complete the backpropagation of the error signal, the content of the register in Fig. 3 is also fed to a channel estimator. This estimator uses the received signals along with the currently available beamforming codebook to reconstruct a *rough* estimate of the channel. Based on (11), we notice that the output z_n of each combiner $\mathbf{w}_n$ is essentially the projection of the channel $\mathbf{h}$ onto the subspace spanned by the combiner $\mathbf{w}_n$. Thus, we estimate a rough version of the channel through

$$\widehat{\mathbf{h}} = \left(\mathbf{W}^H\right)^{\dagger} \mathbf{z}. \tag{26}$$

This approach does not result in an accurate estimate of the channel, yet it helps the learning process as shown in Appendix B.

VI. Practicality of Proposed Solutions

Both proposed solutions are developed with practicality in mind. They are both geared towards handling different challenges commonly faced in designing mmWave beam codebooks, especially with fully-analog architectures. However, that does not mean they operate in the same way. They approach the codebook learning problem from different angles, as briefly discussed below.

The supervised learning solution relies on explicit channel knowledge and follows a *transparent* leaning approach. It requires the mmWave system to operate with some common environment-independent codebook, like the DFT codebook [7], and during its operation it collects channel information from the surroundings. Such information is used to construct the training dataset ($\mathcal{S}_t$) as described in Section IV-B. Once a dataset is available, the central unit trains the model in the background, and upon the completion of the training phase, the new environment-aware codebook is directly plugged into the system. This method decouples the communication operation from the codebook learning process, and allows the system to function normally until a better codebook is learned. Its main drawbacks, however, are the requirement of accurate (or good quality) channel estimates to construct the training dataset, and the relatively lengthy offline learning process.

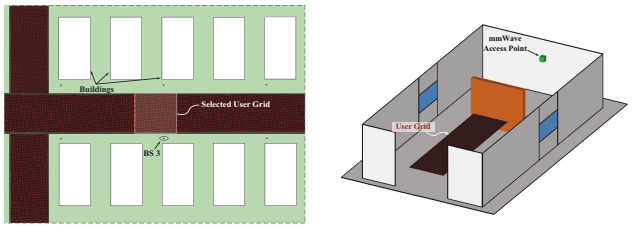
The self-supervised solution, in contrast, trades explicit and accurate channel knowledge for faster training and adaptation. The need for accurate channel estimates in itself is a burden to the mmWave system, especially when hardware impairments are factored in. Hence, the self-supervised solution is designed to transcend that need. As shown in Section V-C, the model is implemented as an integral component of the mmWave system and does not run in the background. The learning instead is performed online while the system is operating. This provides a more adaptable and faster training in terms of implementation. However, this adaptability comes with its own shortcomings. The first one is a subtle degradation in the quality of the learned codebook compared to that of the supervised solution (as will be discussed in Section VIII). It is a direct consequence of implementing a simple yet noisy channel estimator. The other issue is an unstable communication performance at the beginning of the learning process. Different to the transparent nature of the supervised solution, the self-supervised solution learns on the job, and as a results the codebook itself evolves over time.

VII. Experimental Setup and Model Training

In order to evaluate the performance of the proposed codebook learning solutions, two communication scenarios are considered. They are designed to represent two different communication settings. The first has all users experiencing LOS connection with the base station while the other has them experiencing NLOS connection. The following two sections provide more details on the scenarios and the training and testing processes.

A. Communication Scenarios and Datasets

Two communication scenarios are used for performance evaluation. The first one is, as mentioned earlier, a LOS scenario, see Fig. 4-(a). It is an outdoor scene where all users have LOS connection with the mmWave base station. The



(a) LOS Scenario (b) NLOS Scenario

Fig. 4. Two perspective views of the considered communication scenarios. (a) shows the LOS scenario. It is chosen to be outdoor since the likelihood of LOS connection is higher there. (b) shows the NLOS scenario. Similar to (a), this scenario has been chosen for the high likelihood of having NLOS users indoors.

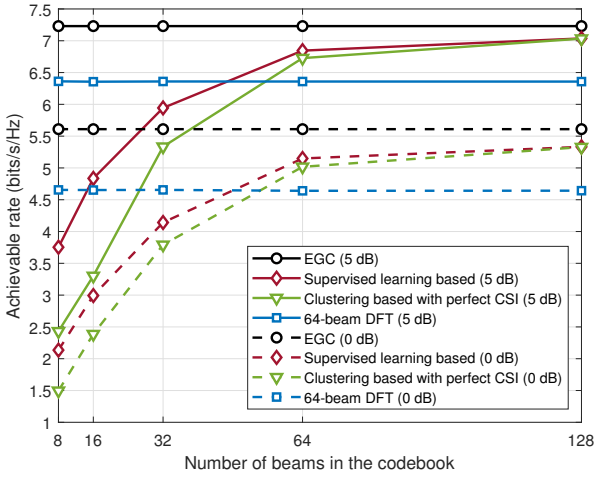
TABLE I
HYPER-PARAMETERS FOR CHANNEL GENERATION

Parameter	value	
Name of scenario	O1_28	I2_28B
Active BS	3	1
Active users	800 to 1200	1 to 700
Number of antennas (x, y, z)	(1, 64, 1)	(64, 1, 1)
System BW	0.2 GHz	0.2 GHz
Antenna spacing	0.5	0.5
Number of OFDM sub-carriers	1	1
OFDM sampling factor	1	1
OFDM limit	1	1
Number of paths	5	5

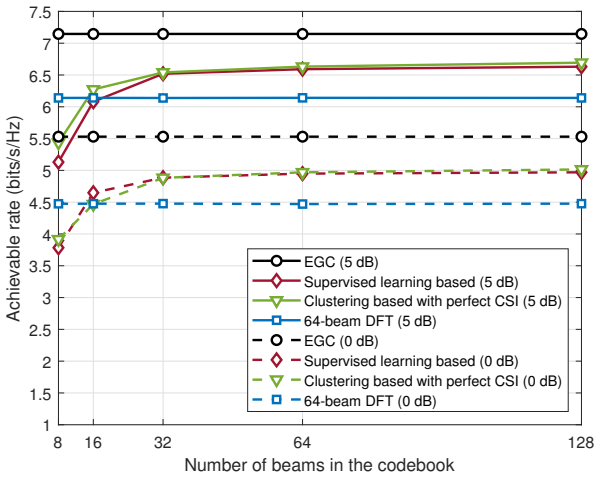
second scenario, on the other hand, is chosen to be an indoor NLOS scenario where all users have NLOS connection with the mmWave base station. Both scenarios are for an operating frequency of 28 GHz, and both are part of the DeepMIMO dataset [24]. Using the data-generation script of DeepMIMO, two sets of channels, namely $\mathcal{S}^{\text{LOS}}$ and $\mathcal{S}^{\text{NLOS}}$, are generated with $|\mathcal{S}^{\text{LOS}}| \approx 73 \times 10^3$ samples and $|\mathcal{S}^{\text{NLOS}}| \approx 150 \times 10^3$ samples. Table I shows the data-generation hyper-parameters. For the supervised solution, both sets undergo processing to generate the labels and create two sets of pairs as described in Section IV-B2. The new datasets are henceforth referred to as $\mathcal{S}^{\text{LOS}}_{t_1}$ and $\mathcal{S}^{\text{NLOS}}_{t_1}$. For the self-supervised solution, on the other hand, labels are not needed, and, therefore, the two sets $\mathcal{S}^{\text{LOS}}$ and $\mathcal{S}^{\text{NLOS}}$ are used as they are. For the sake of convenience, these two sets will be re-named $\mathcal{S}^{\text{LOS}}_{t_2}$ and $\mathcal{S}^{\text{NLOS}}_{t_2}$. To evaluate the performance of the proposed solutions, we will compare them with the DFT codebook; we define the n-th beam of an M-point DFT codebook as $\mathbf{w}^{\text{DFT}}_n = \frac{1}{\sqrt{M}}\left[1, e^{j\frac{2\pi n}{M}}, ..., e^{j\frac{2\pi n(M-1)}{M}}\right]^T, n = 0, 1, ..., M-1$.

B. Model Training

The two models are trained and tested on their datasets introduced in the earlier section, Section VII-A. The training of both solutions follow the same strategy. It starts by data pre-processing. The channels in each dataset are normalized to improve the training experience [33], which is a very common practice in machine learning. As in [37] [38], [39] and [40], the channel normalization using the maximum absolute value in the training dataset helps the network undergo a stable and



(a) LOS scenario



(b) NLOS scenario

Fig. 5. The achievable rate versus the number of beams of the codebook using the supervised solution in: (a) LOS scenario and (b) NLOS scenario. It shows the performance under two receive SNRs, 0 and 5 dB.

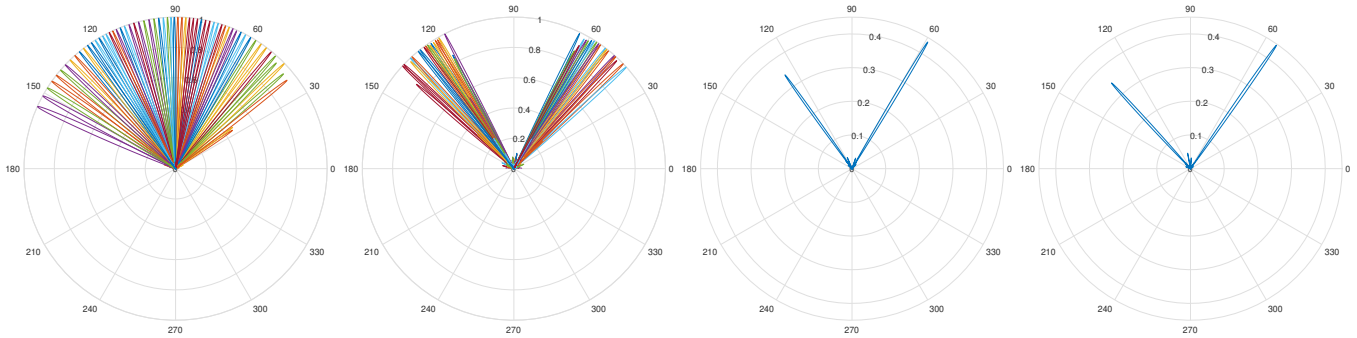
efficient training. Formally, the normalization factor is found as follows

$$\Delta = \max_{\mathbf{h} \in S} |h_{m,u}|^2 \tag{27}$$

where $h_{m,u} \in \mathbb{C}$ is the mth element in the channel vector of the uth user, and $\mathcal{S} \in \{\mathcal{S}^{\text{LOS}}_{t_1}, \mathcal{S}^{\text{NLOS}}_{t_1}, \mathcal{S}^{\text{LOS}}_{t_2}, \mathcal{S}^{\text{NLOS}}_{t_2}\}$. Using the normalized channels, each solution is, then, trained on 70% samples of the dataset selected randomly and validated on the rest. The other training hyper-parameters are: ADAM [41] optimizer, 0.1 learning rate, 500 batch size, and 5 training epochs. Example3 scripts of the developed codebook learning solutions are available in [42] and [43].

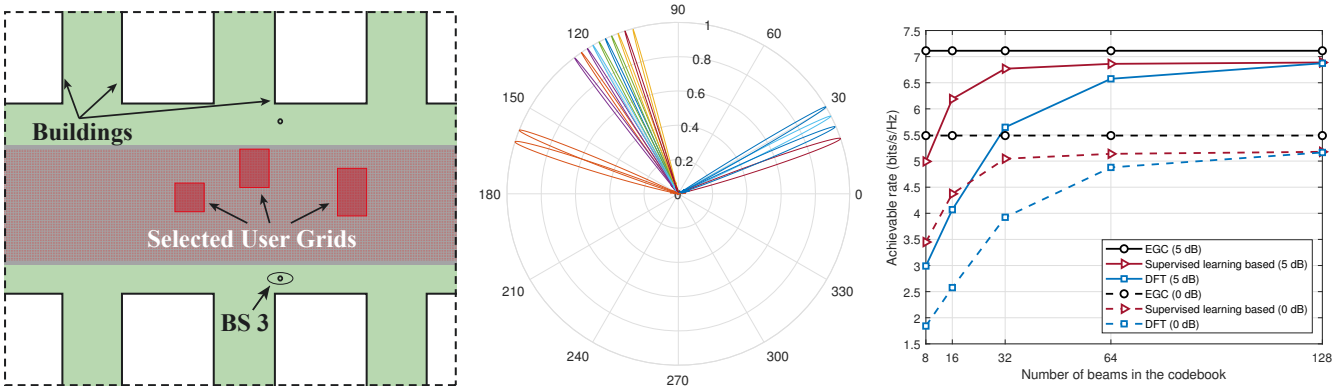
VIII. Simulation Results

In this section, we evaluate the performance of the proposed solutions using the scenarios described in Section VII. The evaluation starts with the supervised solution as it is a stepping stone towards the self-supervised one. The numerical results show that the proposed solutions have clear advantages compared to classical approaches, and they can efficiently adapt



(a) LOS (b) NLOS (c) Single-beam, NLOS (d) Single-beam, NLOS

Fig. 6. Beam patterns for the learned codebook using the supervised solution. (a) shows the codebook learned for the LOS scenario while (b) shows that learned for the NLOS scenario. Two beams from the 64-beam NLOS codebook are singled out in (c) and (d). They clearly show that the proposed solution is capable of learning multi-lobe beams.



(a) Non-uniform user distribution (b) Learned 16-beam codebook (c) Performance vs. codebook size

Fig. 7. Performance of the proposed solution on an outdoor scenario where the users are non-uniformly distributed. (a) shows the selected user grids on the adopted outdoor scenario. (b) shows the learned 16-beam codebook. (c) shows the performance versus codebook size and the comparisons with EGC and DFT.

to different environments, user distributions, as well as array imperfections.

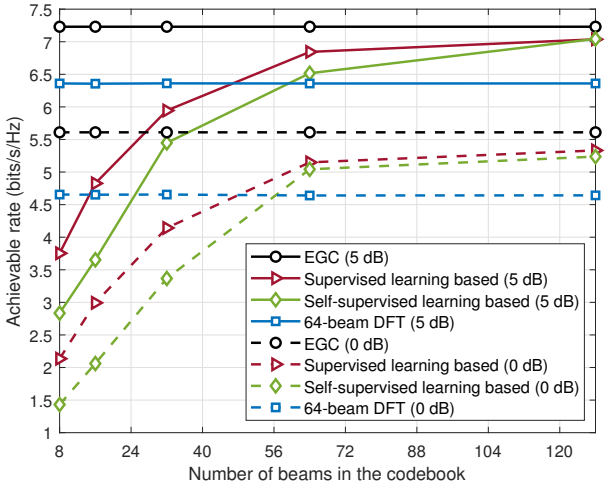
A. Simulation Results for the Supervised Solution

The performance of the supervised solution is investigated from different perspectives as shown below.

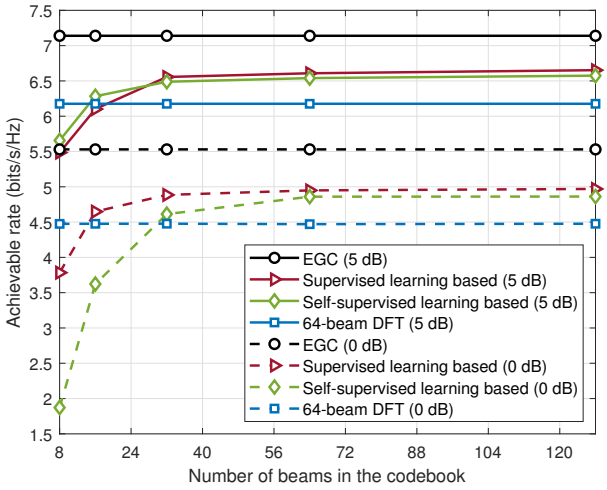
1) Performance in a LOS setting: Fig. 5(a) shows the achievable rate versus the codebook size under 0 dB and 5 dB SNRs for the proposed solution and a classical clustering-based approach that has been introduced in [44] (henceforth referred to as the classical approach). The learned codebook exhibits interesting behavior compared to a 64-beam DFT codebook, an EGC receiver, as well as the classical approach. With half the number of beams of a DFT codebook, the learned codebook achieves more than 80% of the rate that the DFT achieves and displays a clear advantage over the classical approach. This is very important as smaller codebook sizes mean less beam training overhead. Further, the figure shows that when the learned codebook is allowed to grow in size and match the size of the DFT codebook, its performance surpasses that of the DFT codebook. This is quite intriguing as, typically, a DFT codebook performs very well in a LOS setting. This observation applies equally to the classical approach. Its performance improves and gets closer to that of the proposed solution with larger codebook sizes.

2) Performance in a NLOS setting: In such setting, a codebook should be able to capture as much power as possible, and, to do that, it might need to form multi-lobe beams. Similar to the LOS case, Fig. 5(b) depicts the achievable rate of the proposed solution and the classical approach versus the number of beams. Two interesting observations to point out here: (i) both learned codebooks, with supervised and classical solutions, outperform the 64-beam DFT codebook with a fraction of the number of beams; and (ii) the supervised and classical codebooks exhibit similar performance. The first observation is a direct consequence of the ability of both solutions to adapt to the environment and users' channels. The second one, on the other hand, could be explained by the geometry of the environment, see Fig. 4(b). As it will be discussed in the next subsection and Fig. 6(b), the geometry of the environment allows for two narrow angle spreads at the base station, which are quite different in direction. This is argued to result in two distinct and dense master clusters that could readily be uncovered.

3) What does the learned codebook look like: To develop a deeper understanding of the performance of the proposed



(a) LOS scenario



(b) NLOS scenario

Fig. 8. The achievable rate vs. number of beams in the codebook with supervised and self-supervised learning solutions in: (a) a LOS scenario and (b) NLOS scenario. Both figures have the results for 0 dB and 5 dB receive SNR.

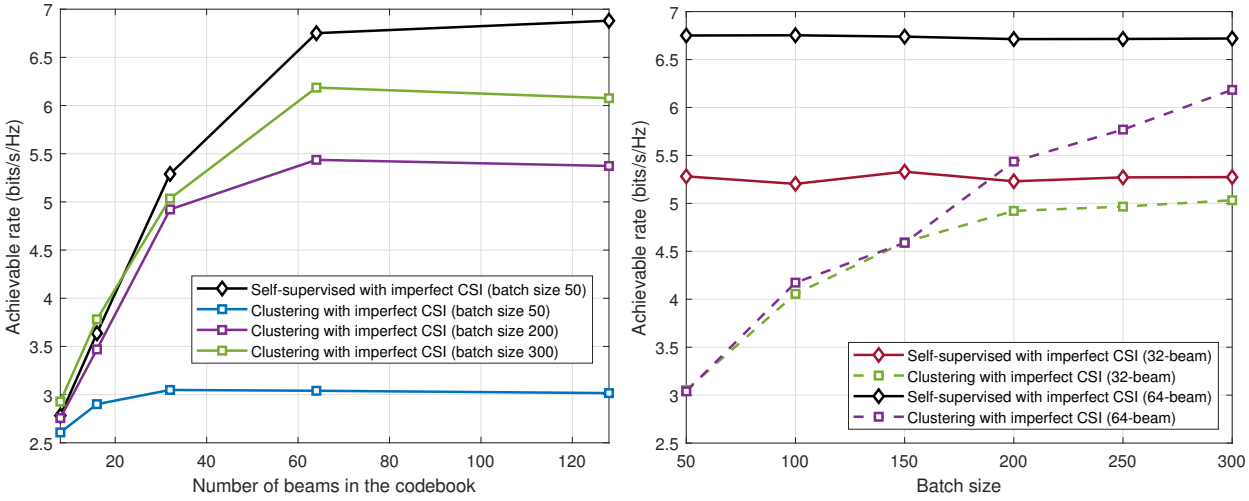
solution and verify its capability of learning beams that adapt to the surrounding environment and user distributions, the resulting beam patterns is plotted in Fig. 6. More specifically, this figure shows different beam patterns of two different 64-beam codebooks learned in LOS and NLOS settings. The patterns in Fig. 6(a) are for the LOS case, and they explain the improvement the learned 64-beam codebook experiences compared to the DFT codebook; all the learned beams are directive and have single-lobe, yet they do not spread across the whole azimuth plane like the DFT beams do. Their spread, instead, follows the user distribution in the scenario, the red rectangle drawn in Fig. 4(a). This makes each beam in the codebook tuned to serve a certain group of users and none of the beams is "wasted" by any means. In the NLOS setting, Fig. 6(b) shows how the solution captures the different NLOS paths in the environment; the codebook is almost evenly split between the two major scatterers, the two side walls of the room. As a matter of fact, **looking at Fig. 6(c) and Fig. 6(d) reveals that the learned beams are not exclusively single-lobe, as some beams have multiple lobes that adapt to the main scatterers in the room**. This is a quite important property for a NLOS beam codebook, and it is evident in the codebook performance in Fig. 5(b). It explains the clear gap in performance between the learned and DFT codebooks, and it hints to the reason why both the supervised and classical solutions have similar performance.

4) Demonstration of the user distribution awareness: To better demonstrate the user distribution awareness of the proposed solution, we generate another dataset where the users are non-uniformly distributed across the space, as shown in Fig. 7(a). In Fig. 7(b), we show the learned 16-beam codebook, where all the beams are pointing towards the users and there is no beam wasted on the places where there is no user at all. As a result, the learned codebook is able to outperform DFT codebook with much smaller codebook size, as shown in Fig. 7(c), which is tantamount to the significant reduction on the latency incurred by beam training.

B. Simulation Results for the Self-supervised Solution

A good starting point for the discussion about the self-supervised solution is to benchmark it to its supervised counterpart. Fig. 8 does exactly that by plotting the achievable rate versus the number of beams for both solutions in a LOS and NLOS settings and under 5 dB SNR. Fig. 8(a) shows that the self-supervised solution is slightly lagging behind the supervised one; the self-supervised solution achieves over 90% and 95% of the achievable rates obtained by the supervised solution at 32 beams and 64 beams, respectively, and the gap between the two solutions keeps shrinking as larger codebooks are learned. In a NLOS setting, on the other hand, the gap between the two almost disappears as shown in Fig. 8(b). Similar to the discussion in Sections VIII-A2 and VIII-A3, this could be attributed to the geometry of the environment. **Overall, the results in both settings are very intriguing and promising, for the self-supervised solution achieves this performance without explicit channel knowledge.**

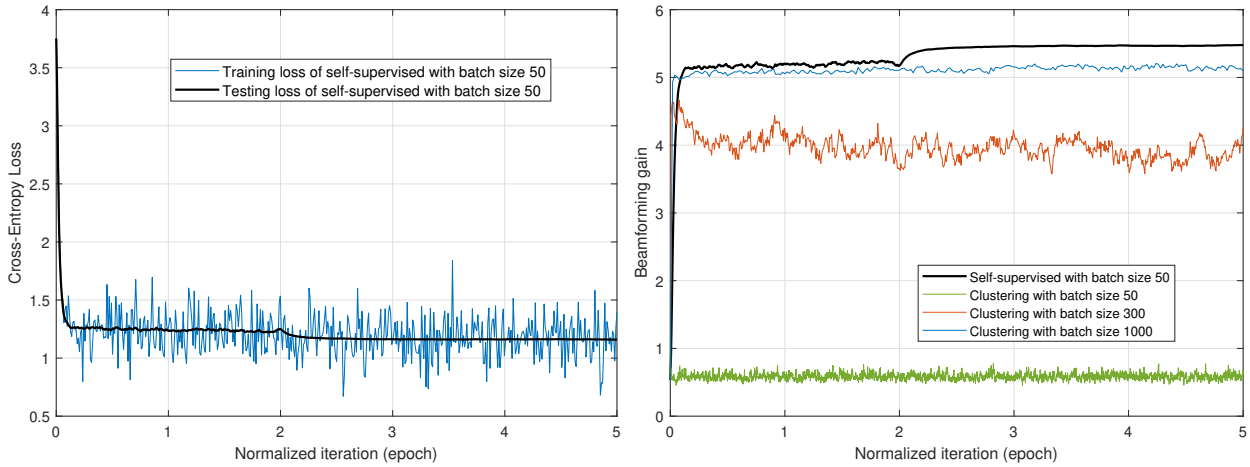
1) Online learning: Learning while the wireless system is operating (i.e., online learning) is an important property for the proposed self-supervised solution, as discussed in Section VI. To demonstrate the value of such property, the proposed solution is compared to the classical approach in an online fashion with imperfect CSI, i.e., $\widehat{\mathbf{h}}$ in (26). Multiple vectors of combined received signals (11) from various users are accumulated to form a batch, on which each solution is trained. Fig. 9(a) shows that when the batch size is set to 50, the self-supervised solution significantly outperforms the classical solution. As the batch size is allowed to grow, the performance of the classical approach improves. It gets close to that of the proposed one with batch sizes that are greater than or equal to 300. For the self-supervised solution, however, batch size has less impact; Fig. 9(b) shows how the self-supervised solution maintains a very balanced performance under different batch sizes. The observations from both figures could be attributed to how the proposed solution learns from a batch, which will be discussed at the end of the next subsection (Section VIII-B2).



(a) Achievable rate vs. codebook size

(b) Achievable rate vs. batch size

Fig. 9. The achievable rate under different batch sizes. Both results are based on LOS scenario and on 5 dB receive SNR.



(a) Convergence behavior (loss function)

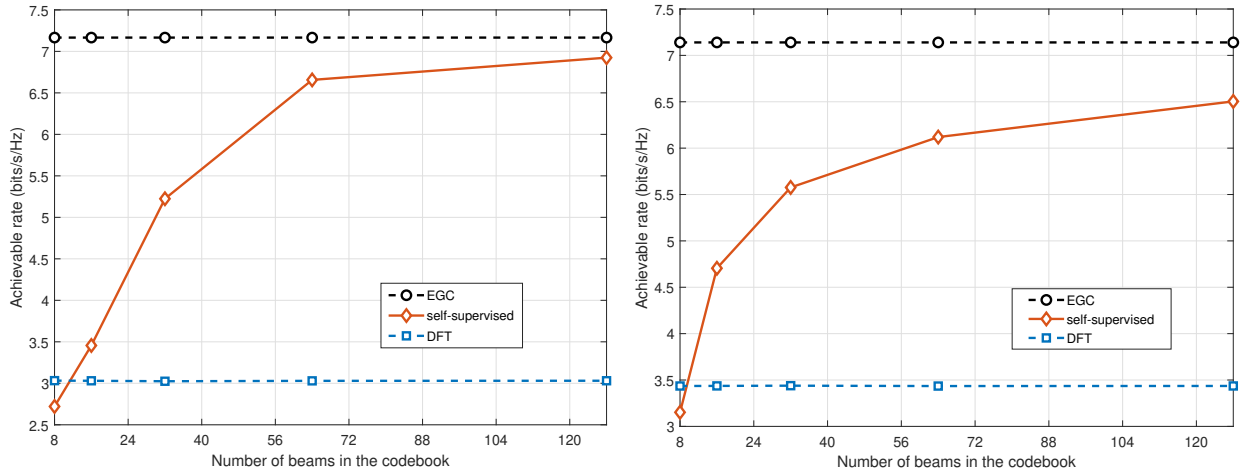
(b) Convergence behavior (beamforming gain)

Fig. 10. The convergence behaviors of the self-supervised approach and the clustering based approach: (a) Training and testing losses of the self-supervised solution and (b) beamforming gain performance of both solutions on learning a 64-beam codebook.

2) Convergence of the self-supervised solution: Algorithm convergence is an important aspect to investigate for an online learning solution like the self-supervised one. Fig. 10(a) shows the learning progress of the self-supervised solution on the task of learning a 64-beam codebook using a batch size of 50. It plots the training and validation losses versus the normalized number of iterations per epoch[1]. The first observation from the figure is how fast the proposed solution converges. With less than 102 iterations (around 0.1 normalized iteration on the x-axis), the proposed solution records a validation loss of ≈ 1.3, only 0.1 away from saturation. This is translated into wireless communication terms in Fig. 10(b), where the solution achieves over 90% of the beamforming gain of the final codebook with 102 training iterations.

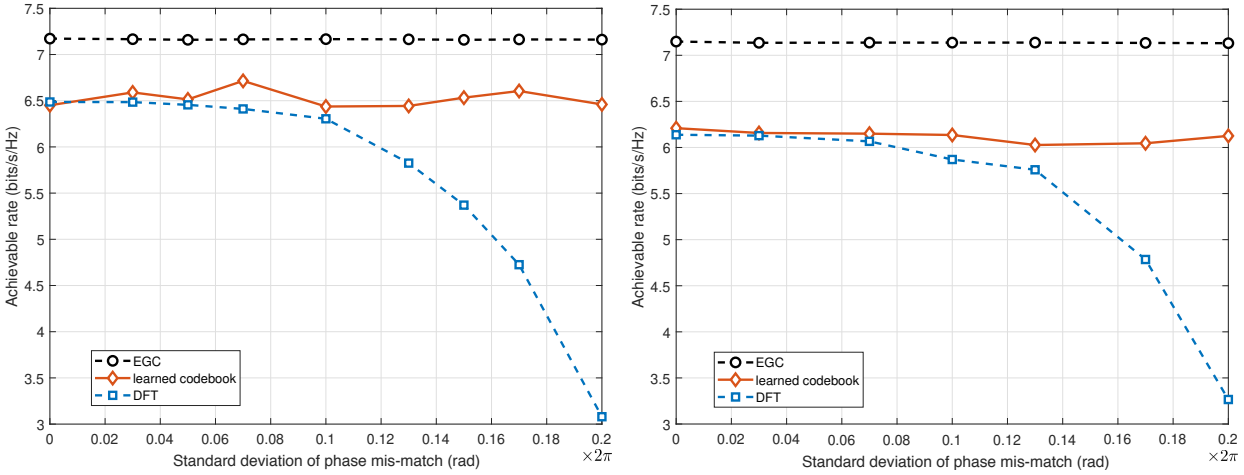
Fig. 10(b) also benchmarks the convergence of the proposed solution to that of the classical one on the validation set. The comparison is done using different batch sizes for the classical approach. With a batch size of 50, the beamforming gain of the classical approach is noisy and does not converge to anything meaningful. As the batch size increases, the approach starts exhibiting a sensible behavior, producing meaningful codebooks. This is the case because **the classical approach relies on a k-means-like algorithm that updates all centroids (i.e., beams) from a single batch.** Small batches, statistically speaking, do not convey enough information about the wireless channels in an environment as large batches do, which explains the improvement in convergence observed in Fig. 10(b) as well as the improvement in performance observed earlier in Fig. 9(b). The situation is different for the proposed solution; **it only updates beams that are relevant to the combined received-signal vectors in the batch and not all beams in the codebook**, which makes it more stable in terms of performance and convergence.

[1]Iteration number divided by total number of iterations per epoch.



(a) LOS scenario (b) NLOS scenario

Fig. 11. The achievable rate versus number of beams for the self-supervised solution. The performance is evaluated under 5 dB SNR, antenna spacing mismatch with $\sigma_d = 0.1\lambda$ standard deviation, and phase mismatch with $\sigma_p = 0.4\pi$. (a) shows the performance in LOS setting while (b) considers NLOS setting.



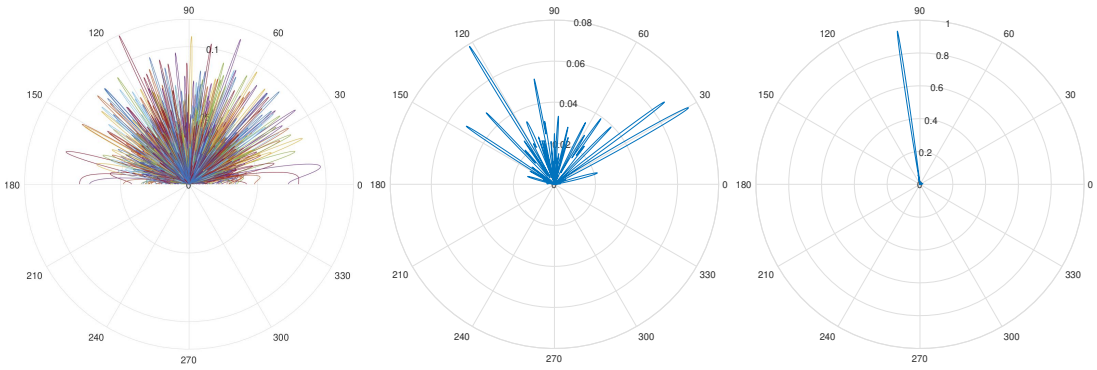
(a) LOS scenario (b) NLOS scenario

Fig. 12. The achievable rate vs. the standard deviation of phase mismatch with self-supervised solution in: (a) LOS and (b) NLOS settings. The performance is evaluated under 5dB receive SNR, antenna spacing mismatch with standard deviation of 0.1λ, and a 64-beams codebook.

C. Performance Evaluation Under Hardware Impairments

The performance of the self-supervised solution is evaluated under hardware impairments using the model introduced in Section II-C. The channels in datasets $\mathcal{S}_{t_2}^{\text{LOS}}, \mathcal{S}_{t_2}^{\text{NLOS}}$, are corrupted with antenna spacing and phase mismatches that, respectively, have $\sigma_d = 0.1\lambda$ and $\sigma_p = 0.4\pi$ standard deviations. Fig. 11(a) shows the simulation result of the proposed solution in a LOS setting. It maintains a similar performance to that presented in Fig. 8(a) and displays an intriguing ability to combat the challenges imposed by the hardware impairments. **This indicates that the self-supervised solution can efficiently adapt to the corrupted (and arbitrary) array-response vectors, compensating for the antenna-spacing and phase mismatches**. The same performance can also be observed in the NLOS case. With the same hardware impairment settings, Fig. 11(b) depicts the achievable rate versus the codebook size in a NLOS setting. The learned codebook continues to maintain the same trend as that in the LOS case. In fact, with 128 beams, the codebook learned can attain over 90% of the achievable rate of the upper bound. Such ability is lacking in classical beam steering codebooks such as the DFT codebook. Compared to Fig. 8(a) and Fig. 8(b), the performance of DFT codebooks degrades significantly when impairments are present. The reason lies in the patterns of the corrupted array response vectors, which lose their directivity and experience critical distortion.

It is important at this stage to pose the following question: How robust is the self-supervised solution? The answer to that question would provide some perspective on the capacity of the proposed solution to endure hardware impairments. In Fig. 12(a), we plot the achievable rates versus the standard deviation of the phase mismatch. The figure considers a LOS setting and a fixed antenna spacing mismatch with a standard deviation of 0.1λ. As the standard deviation of the



(a) Learned 64-beam codebook (b) Projection on clean space (c) Projection on corrupted space

Fig. 13. Beam patterns for codebook with 64 beams learned by the self-supervised solution in LOS setting with hardware impairments (antenna spacing and phase mismatches with, respectively, 0.1λ and 0.4π standard deviations). (a) shows all 64 beams (if plotted for a uniform array), (b) shows one of the codebook beams (if plotted for a uniform array), and, finally, (c) shows the same beam in (b) when plotted for the corrupted array (i.e., the actual beam pattern out of the corrupted array).

phase increases, the self-supervised solution keeps a balanced performance. The DFT codebook, on the other hand, degrades drastically as the level of corruption increases. **This behavior demonstrates the robustness of the proposed codebook learning approach and its ability to adapt to the various hardware impairments.** The same test with the same antenna spacing mismatch is repeated but in the NLOS setting, and the performance is shown in Fig. 12(b). The proposed solution exhibits a similar performance to that in the LOS setting, which further emphasizes the conclusion on its robustness.

To visualize what the proposed solution is learning exactly, Fig. 13 plots different beam patterns from a learned codebook with hardware impairments and in a LOS setting. The first figure on the left, namely Fig. 13(a), shows all beam patterns in the learned codebook when projected on the angular space of the *uniform* (uncorrupted) arrays. One of those beams in plotted again separately in Fig. 13(b). While these beams appear distorted with multiple lobes, they actually look this way because they match the hardware impairments and mismatches. To prove that, we plotted the selected beam again in Fig. 13(c) when projecting it on the angular space of the corrupted beams. In other words, this is the actual far-field beam pattern that the corrupted array will generate. This beam is clearly depicting the supposed pattern, which is a single-lobe beam pointing to the user's direction. All that verifies the interesting capability of the proposed solution in learning beams that adapt to the surrounding environment and given hardware.

IX. Conclusion

In this paper, we considered hardware-constrained mmWave massive MIMO systems and developed machine learning frameworks that learn environment and hardware aware beam codebooks. Achieving that was through designing novel complex-valued neural network architectures that use the neuron weights to directly model the beamforming weights of the phase shifters. Further, these architectures account for the key hardware constraints such as the constant-modulus and quantized-angles constraints. The proposed model is trained online in a self-supervised manner, avoiding the need for explicit channel state information. The developed approach was extensively evaluated using DeepMIMO, in both LOS and NLOS environments. Simulation results show that developed solutions can learn codebook beams that adapt to the surrounding environment and user distribution, which can significantly reduce the training overhead and improve the achievable data rates. When benchmarked to a classical clustering-based approach, the self-supervised solution shows clear advantages over the classical one in terms of convergence and online learning. Furthermore, the results demonstrate the capability of the proposed self-supervised solution in adapting the beam patterns to the hardware impairments and unknown array geometry. This highlights the potential gains of leveraging machine learning to develop deployment- and hardware-aware beamforming codebooks.

Appendix

A. Complex Differentiability

The problem with (19) and (25) lies in their complex differentiability, more specifically, the complex differentiability of $\frac{\partial \mathbf{q}}{\partial \mathbf{z}}$ and $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}$. We refer to the work of [45] where an argument is presented to circumvent this limitation. It states that in order to perform backpropagation in a complex-valued neural network, a sufficient condition is to have a cost function and activations that are differentiable with respect to the real and imaginary parts of each complex parameter in the network. Formally, let $w = w^{\mathrm{r}} + jw^{\mathrm{im}} \in \mathbb{C}$ and $z = f(w) \in \mathbb{R}$ such that it does not satisfy Cauchy-Riemann equations. In this case, z is not complex differentiable, and the suggested way around this problem is to view w^{r} and w^{im} as two independent variables such that $w^{\mathrm{r}},\ w^{\mathrm{im}} \in \mathbb{R}$. Then, the "gradient" of z is defined as

$$\nabla z = \left[\frac{\partial}{\partial w^{\mathrm{r}}} f(w), \frac{\partial}{\partial w^{\mathrm{im}}} f(w) \right]^T. \tag{28}$$

For instance, if $z = (w^{\mathrm{r}})^2 + (w^{\mathrm{im}})^2 \in \mathbb{R}$, then

$$\nabla z = \left[\frac{\partial}{\partial w^{\mathrm{r}}} \left[(w^{\mathrm{r}})^2 + (w^{\mathrm{im}})^2 \right], \frac{\partial}{\partial w^{\mathrm{im}}} \left[(w^{\mathrm{r}})^2 + (w^{\mathrm{im}})^2 \right] \right]^T, \tag{29}$$

$$= 2 \left[w^{\mathrm{r}}, w^{\mathrm{im}} \right]^T. \tag{30}$$

B. Computing the Partials

Going back to (19) and (25), the factors $\frac{\partial \mathbf{q}}{\partial \mathbf{z}}$ and $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}$ satisfy the condition, and hence, we construct the Jacobian $\frac{\partial \mathbf{q}}{\partial \mathbf{z}}$ as

$$\frac{\partial \mathbf{q}}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial q_1}{\partial z_1^{\mathrm{r}}} & 0 & \dots & 0 & \frac{\partial q_1}{\partial z_1^{\mathrm{im}}} & 0 & \dots & 0 \\ 0 & \frac{\partial q_2}{\partial z_2^{\mathrm{r}}} & \dots & 0 & 0 & \frac{\partial q_2}{\partial z_2^{\mathrm{im}}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial q_N}{\partial z_N^{\mathrm{r}}} & 0 & 0 & \dots & \frac{\partial q_N}{\partial z_N^{\mathrm{im}}} \end{bmatrix}. \tag{31}$$

The sparsity of the Jacobian follows from the fact that $\mathbf{q}$ is the result of an element-wise operation, see (14). The reason behind its shape, i.e., $N \times 2N$, will be explained shortly. Since the output of the n-th combiner z_n is only determined by the n-th column of the matrix $\mathbf{W}$ (see (11)) and since the n-th column of $\mathbf{W}$ is only a function in $\boldsymbol{\theta}_n$ (see (13)), we can write the other Jacobian, namely $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n}$, as

$$\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_n} = \begin{bmatrix} 0 & \dots & \frac{\partial z_n^{\mathrm{r}}}{\partial \theta_{1n}} & \dots & 0 & \dots & \frac{\partial z_n^{\mathrm{im}}}{\partial \theta_{1n}} & \dots & 0 \\ 0 & \dots & \frac{\partial z_n^{\mathrm{r}}}{\partial \theta_{2n}} & \dots & 0 & \dots & \frac{\partial z_n^{\mathrm{im}}}{\partial \theta_{2n}} & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & \frac{\partial z_n^{\mathrm{r}}}{\partial \theta_{Mn}} & \dots & 0 & \dots & \frac{\partial z_n^{\mathrm{im}}}{\partial \theta_{Mn}} & \dots & 0 \end{bmatrix}^T, \tag{32}$$

which has a shape of $2N \times M$. Now, to calculate $\frac{\partial z_n^{\mathrm{r}}}{\partial \theta_{mn}}$ or $\frac{\partial z_n^{\mathrm{im}}}{\partial \theta_{mn}}$, $\forall m = \{1, \dots, M\}$, we recall (12) and write z_n^{r} and z_n^{im} as functions of the n-th column of $\mathbf{W}$ as follows

$$z_n^{\mathrm{r}} = \sum_{m=1}^{M} w_{mn}^{\mathrm{r}} h_m^{\mathrm{r}} + w_{mn}^{\mathrm{im}} h_m^{\mathrm{im}}, \tag{33}$$

$$z_n^{\mathrm{im}} = \sum_{m=1}^{M} \left(-w_{mn}^{\mathrm{im}} \right) h_m^{\mathrm{r}} + w_{mn}^{\mathrm{r}} h_m^{\mathrm{im}}, \tag{34}$$

where

$$w_{mn}^{\mathrm{r}} = \frac{1}{\sqrt{M}} \cos\left(\theta_{mn}\right), \; w_{mn}^{\mathrm{im}} = \frac{1}{\sqrt{M}} \sin\left(\theta_{mn}\right). \tag{35}$$

The partials now are computed as follows

$$\frac{\partial z_n^{\mathrm{r}}}{\partial \theta_{mn}} = \frac{\partial z_n^{\mathrm{r}}}{\partial w_{mn}^{\mathrm{r}}} \cdot \frac{\partial w_{mn}^{\mathrm{r}}}{\partial \theta_{mn}} + \frac{\partial z_n^{\mathrm{r}}}{\partial w_{mn}^{\mathrm{im}}} \cdot \frac{\partial w_{mn}^{\mathrm{im}}}{\partial \theta_{mn}}, \tag{36}$$

$$= \frac{1}{\sqrt{M}} \left(-h_m^{\mathrm{r}} \sin\left(\theta_{mn}\right) + h_m^{\mathrm{im}} \cos\left(\theta_{mn}\right) \right), \tag{37}$$

and

$$\frac{\partial z_n^{\mathrm{im}}}{\partial \theta_{mn}} = \frac{\partial z_n^{\mathrm{im}}}{\partial \left(-w_{mn}^{\mathrm{im}}\right)} \cdot \frac{\partial \left(-w_{mn}^{\mathrm{im}}\right)}{\partial \theta_{mn}} + \frac{\partial z_n^{\mathrm{im}}}{\partial w_{mn}^{\mathrm{r}}} \cdot \frac{\partial w_{mn}^{\mathrm{r}}}{\partial \theta_{mn}}, \tag{38}$$

$$= \frac{1}{\sqrt{M}} \left(-h_m^{\mathrm{r}} \cos\left(\theta_{mn}\right) - h_m^{\mathrm{im}} \sin\left(\theta_{mn}\right) \right). \tag{39}$$

Evaluating (37) and (39) clearly relies on the channel estimates. This should not be a problem for the supervised solution, but for the self-supervised solution, the estimate obtained using (26), namely $\hat{\mathbf{h}}$, is substituted for $\mathbf{h}$.

Having found the partials, the reason behind the choice of the matrix shapes in (31) and (32) could be explained. The final objective of (19) and (25) is to propagate back the error signal and update the parameters of the codebook as in (20). The matrix forms of (31) and (32) guarantees that the computation of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_n}$ could be performed in simple matrix multiplication, which is critical for efficient implementation.

REFERENCES

[1] Y. Zhang, M. Alrabeiah, and A. Alkhateeb, "Learning beam codebooks with neural networks: Towards environment-aware mmwave MIMO," in *Proc. of IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), arXiv e-prints*, 2020, p. arXiv:2002.10663.

[2] Y. Ghasempour and C. R. C. M. da Silva and C. Cordeiro and E. W. Knightly, "IEEE 802.11ay: Next-Generation 60 GHz Communication for 100 Gb/s Wi-Fi," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 186–192, 2017.

[3] J. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. Soong, and J. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[4] M. Giordani, M. Polese, A. Roy, D. Castor, and M. Zorzi, "A tutorial on beam management for 3gpp nr at mmwave frequencies," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 173–196, 2019.

[5] Alkhateeb, A. and Jianhua Mo and Gonzalez-Prelcic, N. and Heath, R.W., "MIMO Precoding and Combining Solutions for Millimeter-Wave Systems," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 122–131, Dec. 2014.

[6] A. Alkhateeb, O. El Ayach, G. Leus, and R. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, Oct. 2014.

[7] S. Hur, T. Kim, D. Love, J. Krogmeier, T. Thomas, and A. Ghosh, "Millimeter wave beamforming for wireless backhaul and access in small cell networks," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4391–4403, Oct. 2013.

[8] IEEE 802.11ad, "IEEE 802.11ad standard draft D0.1." [Online]. Available: www.ieee802.org/11/Reports/tgadupdate.htm

[9] T. Moon, J. Gaun, and H. Hassanieh, "Online millimeter wave phased array calibration based on channel estimation," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019, pp. 1–6.

[10] N. Jindal, "MIMO broadcast channels with finite-rate feedback," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5045–5060, Nov. 2006.

[11] C. K. Au-yeung and D. J. Love, "On the performance of random vector quantization limited feedback beamforming in a MISO system," *IEEE Transactions on Wireless Communications*, vol. 6, no. 2, pp. 458–462, 2007.

[12] K. Huang, R. W. Heath, Jr., and J. G. Andrews, "Limited feedback beamforming over temporally-correlated channels," *IEEE Transactions on Signal Processing*, vol. 57, no. 5, pp. 1959–1975, 2009.

[13] V. Raghavan and V. Veeravalli, "On quantized multi-user beamforming in spatially correlated broadcast channels," in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, June 2007, pp. 2041–2045.

[14] D. J. Love and R. W. Heath, "Limited feedback unitary precoding for spatial multiplexing systems," *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2967–2976, 2005.

[15] V. Raghavan, V. Veeravalli, and A. Sayeed, "Quantized multimode precoding in spatially correlated multiantenna channels," *IEEE Transactions on Signal Processing*, vol. 56, no. 12, pp. 6017–6030, Dec 2008.

[16] J. Lee, J.-K. Han, and J. Zhang, "MIMO technologies in 3GPP LTE and LTE-advanced," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, pp. 3:1–3:10, Mar. 2009.

[17] IEEE 802.11n, "IEEE standard for information technology telecommunications and information exchange between systems local and metropolitan area networks specific requirements part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, p. 12793, 2012.

[18] J. Wang, *et al.*, "Beam codebook based beamforming protocol for multi-Gbps millimeter-wave WPAN systems," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 8, pp. 1390–1399, Nov. 2009.

[19] A. Alkhateeb and R. W. Heath, "Frequency selective hybrid precoding for limited feedback millimeter wave systems," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 1801–1818, May 2016.

[20] Zhang, Jianjun and Huang, Yongming and Zhou, Yu and You, Xiaohu, "Beam Alignment and Tracking for Millimeter Wave Communications via Bandit Learning," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5519–5533, 2020.

[21] Li, Min and *et al.*, "Explore and Eliminate: Optimized Two-Stage Search for Millimeter-Wave Beam Alignment," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4379–4393, 2019.

[22] Zhang, Jianjun and *et al.*, "Codebook Design for Beam Alignment in Millimeter Wave Communication Systems," *IEEE Transactions on Communications*, vol. 65, no. 11, pp. 4980–4995, 2017.

[23] D. Love and R. Heath Jr, "Equal gain transmission in multiple-input multiple-output wireless systems," *IEEE Transactions on Communications*, vol. 51, no. 7, pp. 1102–1110, 2003.

[24] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. of Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb 2019, pp. 1–8. [Online]. Available: http://www.DeepMIMO.net

[25] M. Alrabeiah and A. Alkhateeb, "Deep Learning for TDD and FDD Massive MIMO: Mapping Channels in Space and Frequency," *arXiv e-prints*, p. arXiv:1905.03761, May 2019.

[26] X. Li and A. Alkhateeb, "Deep learning for direct hybrid precoding in millimeter wave massive mimo systems," in *Proc. of Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 800–805.

[27] P. Pal and P. P. Vaidyanathan, "Nested arrays: A novel approach to array processing with enhanced degrees of freedom," *IEEE Transactions on Signal Processing*, vol. 58, no. 8, pp. 4167–4181, Aug 2010.

[28] M. Rubsamen and A. B. Gershman, "Direction-of-arrival estimation for nonuniform sensor arrays: From manifold separation to fourier domain music methods," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 588–599, 2009.

[29] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[30] D. Love, R. Heath, V. Lau, D. Gesbert, B. Rao, and M. Andrews, "An overview of limited feedback in wireless communication systems," *IEEE Journal on Selected Areas in Commun.*, vol. 26, no. 8, pp. 1341–1365, Oct. 2008.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[32] S. S. Haykin *et al.*, *Neural networks and learning machines/Simon Haykin.* New York: Prentice Hall,, 2009.

[33] Y. A. LeCun, *et al.*, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[34] F. Haslinger, *Complex Analysis: A Functional Analytic Approach*. Walter de Gruyter GmbH & Co KG, 2017.

[35] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[37] M. Alrabeiah and A. Alkhateeb, "Deep learning for TDD and FDD massive MIMO: mapping channels in space and frequency," *CoRR*, vol. abs/1905.03761, 2019. [Online]. Available: http://arxiv.org/abs/1905.03761

[38] A. Taha, M. Alrabeiah, and A. Alkhateeb, "Enabling large intelligent surfaces with compressive sensing and deep learning," *arXiv preprint arXiv:1904.10136*, 2019.

[39] Y. Zhang, M. Alrabeiah, and A. Alkhateeb, "Deep learning for massive MIMO with 1-bit ADCs: When more antennas need fewer pilots," *IEEE Wireless Communications Letters*, 2020.

[40] 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. 37 328–37 348, 2018.

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

[42] M. Alrabeiah. [Online]. Available: https://github.com/malrabeiah/learningCB

[43] Y. Zhang. [Online]. Available: https://github.com/YuZhang-GitHub/CBL_Self_Supervised

[44] Alkhateeb, Ahmed and Heath, Robert W., "Frequency Selective Hybrid Precoding for Limited Feedback Millimeter Wave Systems," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 1801–1818, 2016.

[45] C. Trabelsi, *et al.*, "Deep Complex Networks," *arXiv e-prints*, p. arXiv:1705.09792, May 2017.