Deep Learning for SVD and Hybrid Beamforming

Ture Peken, Sudarshan Adiga, *Student Member, IEEE*, Ravi Tandon, *Senior Member, IEEE*, and Tamal Bose

***Abstract*— Hybrid beamforming (BF), which divides BF operation into radio frequency (RF) and baseband (BB) domains, will play a critical role in MIMO communication at millimeter-wave (mmW) frequencies. In principle, we can obtain unconstrained (optimum) beamformers of a transceiver, which approach the maximum achievable data rates, through its singular value decomposition (SVD). Due to the use of finite-precision phase shifters, combined with power constraints, additional challenges are imposed on the problem of designing hybrid beamformers. Motivated by the recent success of machine learning (ML) techniques, particularly in areas such as computer vision and speech recognition, we explore if ML techniques can be effectively used for SVD and hybrid BF. To this end, we first present a data-driven approach to compute the SVD. We propose three deep neural network (DNN) architectures to approximate the SVD, with varying levels of complexity. The methodology for training these DNN architectures is inspired by the fundamental property of SVD, i.e., it can be used to obtain low-rank approximations. We next explicitly take the constraints of hybrid BF into account (such as quantized phase shifters, power constraints), and propose a novel DNN based approach for the design of hybrid BF systems. To validate the DNN based approach, we present simulation results for both approximating the SVD as well as for hybrid BF. Our results show that DNNs can be an attractive and efficient solution for estimating SVD in a data-driven manner. For the simulations of hybrid BF, we first consider the geometric channel model. We show that the DNN based hybrid BF improves rates by up to $50-70\%$ compared to conventional hybrid BF algorithms and achieves $10-30\%$ gain in rates compared with the state-of-art ML-aided hybrid BF algorithms. We also discuss the impact of the choice of hyperparameters, such as the number of hidden layers, mini-batch size, and training iterations on the accuracy of DNNs. Furthermore, we provide time complexity and memory requirement analyses for the proposed approach and state-of-the-art approaches.**

***Index Terms*— Hybrid beamforming, singular value decomposition, machine learning, massive MIMO, millimeter-waves.**

I. Introduction

THE mmW band is crucial for enabling fifth-generation (5G), autonomous vehicles, and Internet of Things (IoT) enabled networks [1]–[3]. The combination of abundant bandwidth of mmW frequencies ranging from 30 to 300 GHz, along with a massive number of antennas, has the potential of providing high data rates, improving spectral efficiency, and signal coverage [4]. Since the wavelength gets smaller with the higher frequencies, propagation loss increases in mmWs. Large-scale antenna systems (massive MIMO) can focus the radiated energy toward the specific directions by using directional BF, which compensate for the performance degradation due to the propagation loss [5]. Traditionally, BF has been performed either in RF or BB domain. Digital BF provides higher data rates; however, it requires a large number of RF chains, leading to increased power consumption and cost [4]. On the other hand, analog BF requires fewer RF chains, thus lower power consumption; however, compromises on the achievable rate [6]. The key idea behind hybrid BF is to combine analog BF and digital BF, with the ultimate goal of keeping the power consumption low, and data rates high [7]. Optimal unconstrained beamformers (which maximize channel capacity) can be found through the SVD of the channel, i.e., k singular vectors corresponding to the largest singular values of the channel matrix can be used to determine k optimum beam directions. On the other hand, analog and digital beamformers in hybrid BF must be designed jointly to approach the maximum achievable rates by considering the constraints due to the use of finite-precision phase shifters in the RF domain along with the power constraint. In particular, the elements of RF beamformers are constrained to have constant modulus and quantized phase values. As the RF and digital beamformers in hybrid BF are required to be designed jointly and repeatedly in real-time with changing channel conditions, the selection of beamformers with maximum achievable rates becomes a challenging task.

Several methods have been proposed for hybrid BF design in the literature. In [8], the beams are selected exhaustively based on maximum signal-to-noise-ratio (SNR). However, choosing the beams by using exhaustive search methods leads to high computational complexity. Near-optimum algorithms using sparse approximation techniques have been proposed for hybrid BF as well [9]–[11]. Even though sparse approximation methods can reduce the computational complexity compared to exhaustive techniques, they still need significant training overhead that scales with the number of antennas. Hybrid BF techniques for multi-user MIMO (MU-MIMO) systems have also been proposed in [12], [13]. A low-complexity two-stage multi-user hybrid BF algorithm, which assumes the availability of a limited feedback channel, has been presented in [12]. Authors of [13] propose a hybrid BF algorithm for multi-user massive MIMO systems, which determines the beamformers using weighted sum mean square error (WMSE) minimization. There has been significant recent interest in exploring the use of ML techniques for the design of wireless systems [14]–[18].

Manuscript received July 22, 2019; revised February 28, 2020; accepted June 13, 2020. Date of publication June 30, 2020; date of current version October 9, 2020. This work was supported in part by the Broadband Wireless Access and Applications Center (BWAC) and in part by NSF Award 1822071. The work of Ravi Tandon was supported in part by NSF under Grant CAREER 1651492 and Grant CNS 1715947 and in part by the 2018 Keysight Early Career Professor Award. The associate editor coordinating the review of this article and approving it for publication was E. De Carvalho. *(Corresponding author: Ravi Tandon.)*

The authors are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721 USA (e-mail: turepeken@email.arizona.edu; adiga@email.arizona.edu; tandonr@email.arizona.edu; tbose@email.arizona.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2020.3004386

In [14], the authors study the use of convolutional neural networks (CNNs) for the problem of modulation classification. In [15], the design of single-user MIMO (SU-MIMO) systems is considered through unsupervised learning using an autoencoder. In this scheme, different communication tasks such as modulation and encoding are combined into a single end-to-end system. In [16], long short-term memory (LSTM)-aided nonorthogonal multiple access (NOMA) scheme is proposed to detect channel characteristics. Authors of [17] propose to integrate deep learning (DL) methods into massive MIMO systems for channel estimation and direction-of-arrival (DoA) estimation by employing DNNs. In [18], the development of DL-based solutions for 5G communications has been reviewed, and then novel schemes for DL-based 5G scenarios have been introduced.

ML approaches have also been recently explored for the SVD and BF. Since the computation of SVD by conventional methods such as [19], [20] requires intensive time for large matrices, authors of [21] propose a linear NN to compute the SVD in real-time [21]. In [22], an autoencoder is introduced to find the truncated SVD of a given matrix. A DL-based fully-digital BF design has been proposed in [23]. The authors of [23] divide BF design problem into power allocation and virtual BF design, and then propose the BF prediction network for power allocation and predicting beamformers. In [24], a DNN-based BF approach has been presented to learn the optimum beamformers, which maximize the spectral efficiency under hardware constraint with imperfect channel state information (CSI). In [25], an adaptive cross-entropy (CE) optimization has been proposed for a switch and inverter (SI)-based hybrid precoding architecture. However, the achievable sum-rate needs to be calculated for all the candidate beamformers, which still brings a significant computational overhead. In [26], beam selection in hybrid BF has been considered as a multi-class classification problem. The authors of [26] have adopted the support vector machine (SVM) algorithm to select beamformers that maximize the sum rate over the mmW channel. A DL model, which predicts the BF vectors at the base stations (BS) by using received pilot signals, has been proposed in [27]. The main idea in this method is to use the received signals with omni beam patterns to learn the RF-BF vectors. After RF-BF vectors are selected, BB beamformers are designed by using maximum ratio combining (MRC) technique. A DL-based mmW massive MIMO for hybrid BF is presented in [28]. In this paper, an autoencoder is used to estimate the analog and digital precoders by adopting geometric mean decomposition (GMD) technique. Authors of [29] propose a hybrid BF scheme relying on ML assisted link adaptation. This scheme selects either spatial multiplexing or diversity-aided transmission based on different channel conditions. In [30], first a novel technique to generate datasets for mmW MIMO scenarios has been presented. Then, DL is leveraged for beam-selection using the generated datasets.

The main idea of this work is to formulate the hybrid BF as a constrained SVD problem since the SVD based unconstrained BF constructs an upper bound on the maximum achievable rates. Motivated by the recent success of ML in applications such as image and speech processing [31], [32], we aim to study the potential of ML approaches for the SVD and hybrid BF. The complexity of the conventional SVD algorithms increases quadratically with the dimension of the matrix, and our motivation for calculating the SVD with ML-based techniques is to reduce this complexity. Moreover, the common property among the different conventional SVD algorithms such as [19], [20] is that they first diagonalize the input matrix by plane rotations and then calculate iteratively singular values and singular vectors of the resulting matrix. Therefore, it is intuitive to use a NN, where the elements of the rotation matrices are the weights to be learned by the NN. Authors of [21] show the validity of using a linear NN for computing the SVD [21]. However, linear NNs have a limited capacity to learn the singular values and singular vectors of large matrices since the SVD is a non-linear operation. Therefore, DNNs are more promising for computing the SVD of large matrices at the expense of higher computational power requirements. Authors of [22] indicate the potential of unsupervised approaches such as autoencoders for computing the SVD and principal component analysis (PCA). It has been shown in [22] that the optimal weight matrix of the linear autoencoder with a squared error loss function is the orthogonal projection onto space spanned by the eigenvectors of the covariance matrix of the input. However, the eigenvectors can be found applying some orthogonalization techniques such as Gram-Schmidt to the weight matrix of the autoencoder, which increases the computational complexity.

In this work, we use CNNs to implement our proposed DNN architectures to compute the SVD even though other approaches like feedforward NNs and recursive neural networks (RNNs) can also be used. The first multilayered CNNs have been proposed in [33] for handwritten digit recognition, and since then have been used successfully in various applications, which involve 2D data processing such as image classification. It has also been shown that CNNs were easier to train than the feedforward fully connected NNs [34]. On the other hand, convolution operations have high computational complexity, which causes the CNNs to be slower than the feedforward NNs. The implementation of CNNs using graphical processing units (GPUs) compensates for the computational complexity issue of CNNs, which makes CNNs more advantageous than the feedforward NNs overall. RNNs have been successfully applied to sequence prediction problems such as speech recognition, human motion prediction, etc. [32], [35]. However, it has been recently indicated that simple CNNs outperform canonical RNNs across many different tasks and datasets while achieving longer effective memory [36]. Therefore, we use CNNs for the implementation of our proposed approaches for the SVD and hybrid BF.

A. Contributions of This Paper

• We first propose three novel DNN architectures to learn the SVD, which is the fundamental operation for finding the unconstrained optimum beamformers at the transmitter (Tx) and the receiver (Rx). The first architecture predicts k most significant singular values and singular vectors of a given

matrix using a single DNN. By leveraging the structure of SVD, a low-complexity DNN architecture for rank-k matrix approximation is introduced. The second architecture consists of k low-complexity DNNs; each DNN is trained to estimate the largest singular value and corresponding right and left singular vectors of the given matrix. To further simplify the SVD operation, we propose a third architecture for rank-1 matrix approximation, which estimates k singular values and singular vectors using a single DNN recursively. We introduce customized loss functions to train the three DNN architectures. In principle, the DNNs are trained to minimize the Frobenius distance between the real and the estimated rank-k approximations of the matrix while forcing the singular vectors to be orthogonal.

• Then, we propose a novel DNN architecture for hybrid BF by incorporating constraints that are specific to hybrid BF. We consider the case where finite-precision phase shifters are used in the RF domain, which restricts the analog beamformers to have constant modulus and quantized phase values. Therefore, quantization layers are included in the proposed DNN for hybrid BF. However, incorporating quantization brings additional challenges due to the non-differentiability of the discretization operation. In particular, when we use gradient-based optimization methods for training, the quantization layers in DNNs produce zero gradients, which prevents to update the weights. To circumvent this issue, we propose four quantization approaches. In the first approach, we use a combination of step and piece-wise linear functions to approximate the phase quantization operation, which provides non-zero gradients during training. In the second approach, we consider a soft quantization by using a combination of several sigmoid functions with different parameters during both forward as well as backward propagation. In the third approach, we use step function in the forward propagation while incorporating sigmoid functions with different parameters during backward propagation. In the fourth approach, we implement a stochastic quantization approach [37] during forward propagation while replacing with a straight-through estimator [38] during backpropagation. Finally, we satisfy the power constraint through normalization layers in the proposed DNN architecture.

• We provide the time complexity analysis for the proposed DNN architectures for SVD and compare their time complexities with the conventional SVD algorithms. We show that the proposed DNN based approaches have a smaller time complexity than the traditional SVD approaches while the number of transmit and receive antennas increases, and the other parameters remain constant. We present a comprehensive set of simulation results to show the advantages of DNNs for learning SVD and for hybrid BF. We implement three DNN architectures for SVD using CNNs and discuss the impact of mini-batch size, the number of hidden layers, and training iterations size on accuracy. With the geometric channel model, we simulate the proposed DNN based hybrid BF algorithm and compare its rates with the unconstrained BF, three conventional hybrid BF algorithms [9], [10], [39], an ML-aided hybrid BF algorithm based on CE optimization [25], two DL-based hybrid BF algorithms [27], [30], and an autoencoder based hybrid BF algorithm [28]. The results show that the proposed algorithm achieves up to $50-70\%$ and $10-30\%$ gains in rates compared to the conventional hybrid BF approaches and ML-based algorithms, respectively. We also compare the performance of the proposed DNN based SVD approaches with the traditional SVD algorithms in terms of the time complexity and memory requirements. Furthermore, we perform a time complexity analysis and compare the DNN based approach to other state-of-the-art methods.[1]

B. Notation

We use the following notation throughout this paper: $\mathbf{A}$ is a matrix, $\mathbf{a}$ is a vector, a is a scalar, and $\mathcal{A}$ is a set. $|\mathbf{A}|$, $\mathbf{A}^T$, $\mathbf{A}^{-1}$, $\mathbf{A}^*$, $\|\mathbf{A}\|_F$, and $\mathsf{rank}\,(\mathbf{A})$ are the determinant, transpose, inverse, Hermitian (conjugate transpose), Frobenius norm, and rank of $\mathbf{A}$, respectively. $[\mathbf{A}]_{r,:}$ and $[\mathbf{A}]_{:,c}$ are the rth row and cth column of $\mathbf{A}$. $\|\mathbf{a}\|_2$ is the Euclidean norm of $\mathbf{a}$. $\mathsf{diag}(a_1,\ldots,a_n)$ denotes a diagonal matrix with the entries of $a_1,\ldots,a_n$ on its diagonal. $\log_b(x)$ and $\mathbb{E}\,[\cdot]$ denote the logarithm of x to base b and expectation respectively. $\mathbf{I}$ is the identity matrix. e stands for Euler's number and j denotes $\sqrt{-1}$. $\mathcal{N}\left(\mu,\sigma^2\right)$ and $\mathcal{N}\left(\mathbf{m},\mathbf{R}\right)$ are a complex Gaussian random scalar with mean μ and variance σ^2 and a complex Gaussian random vector with mean $\mathbf{m}$ and covariance $\mathbf{R}$, respectively. $\mathbb{R}$ and $\mathbb{C}$ denote the set of real and complex numbers, respectively.

II. Preliminaries: SVD and Hybrid BF

This section presents the preliminaries for SVD and hybrid BF considered in the paper. Then, we formulate optimum and hybrid BF by using unconstrained and constrained SVD, respectively.

A. SVD

Given the matrix $\mathbf{H} \in \mathbb{C}^{N_R\times N_T}$ with rank $r \leq l = \min\{N_T, N_R\}$, there exists (i) a unitary matrix $\mathbf{U}\in\mathbb{C}^{N_R\times N_R}$; (ii) a diagonal matrix $\mathbf{\Sigma} \in \mathbb{C}^{N_R\times N_T}$ with non-negative numbers on its diagonal; (iii) a unitary matrix $\mathbf{V}\in\mathbb{C}^{N_T\times N_T}$ that construct the SVD of $\mathbf{H}$ as,

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{1}$$

where $\mathbf{U}^*\mathbf{U} = \mathbf{I}_{N_R}$, $\mathbf{V}^*\mathbf{V} = \mathbf{I}_{N_T}$, and $\mathbf{\Sigma} = \mathsf{diag}(\sigma_1,\ldots,\sigma_l)$, $\sigma_i > 0$ for $1 \leq i \leq r$, $\sigma_j = 0$ for $l = \min(N_R, N_T) \geq j \geq r+1$. The diagonal elements of $\mathbf{\Sigma}$ are singular values, and the columns of $\mathbf{U}$ and $\mathbf{V}$ are left and right singular vectors of $\mathbf{H}$, respectively.

B. Optimum BF Using Unconstrained SVD

Consider a communication system with N_T and N_R antennas at the Tx and Rx, respectively. We denote the channel matrix of this system by $\mathbf{H} \in \mathbb{C}^{N_R\times N_T}$, which can be decomposed as $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$. We define the precoder at the Tx as $\mathbf{T}\in\mathbb{C}^{N_T\times L}$ and the combiner at the Rx as $\mathbf{R}\in\mathbb{C}^{N_R\times L}$.

[1]Source codes for the experiments are available at: https://www.dropbox.com/sh/v0gs7ba0qq5 × 168/AACyqRoCz5m3fhpF-azkbn3Qa?dl=0

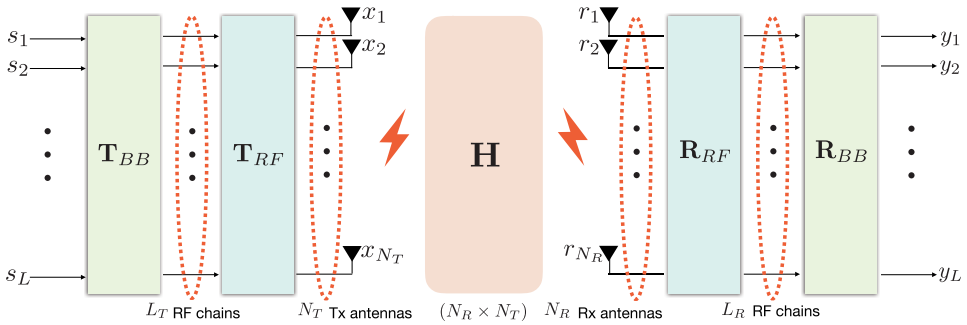


Fig. 1. Hybrid BF architecture with RF and BB blocks. L data streams are processed by the BB precoder $\mathbf{T}_{BB}$. Each BB signal is connected to the one of N_{RF}^T RF chains of the RF precoder $\mathbf{T}_{RF}$. The reverse of this operation is performed at the Rx.

At the Tx, the vector of transmitted symbols $\mathbf{s} \in \mathbb{C}^{L\times 1}$ is first processed by $\mathbf{T}$, and then transmitted from N_T antennas of the Tx. The transmitted signal $\mathbf{x} \in \mathbb{C}^{N_T\times 1}$ is given as,

$$\mathbf{x} = \mathbf{T}\mathbf{s}. \tag{2}$$

Then, N_R antennas of the Rx receive the signal $\mathbf{r} \in \mathbb{C}^{N_R\times 1}$, which is defined as,

$$\mathbf{r} = \mathbf{H}\mathbf{T}\mathbf{s} + \mathbf{n}, \tag{3}$$

where $\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$ is the Gaussian noise vector of dimension $N_R \times 1$. After $\mathbf{r}$ is processed by $\mathbf{R}$, the vector of received symbols $\mathbf{y} \in \mathbb{C}^{L\times 1}$ is obtained as,

$$\mathbf{y} = \mathbf{R}^*\mathbf{H}\mathbf{T}\mathbf{s} + \mathbf{R}^*\mathbf{n}. \tag{4}$$

The optimum beamformers for the given communication system can be found by maximizing some performance utility metric such as SNR [40], achieved rate [41], etc. For the scope of this paper, we focus on the achieved rate, which is maximized by selecting the singular vectors of $\mathbf{H}$ as the beamformers of this system. In particular, the optimum beamformers are found by maximizing the rate R, which is given as,

$$R = \log_2\left(\left|\mathbf{I} + \tfrac{P}{L}\mathbf{C}_n^{-1}\mathbf{R}_{opt}^*\mathbf{H}\mathbf{T}_{opt}\mathbf{T}_{opt}^*\mathbf{H}^*\mathbf{R}_{opt}\right|\right), \tag{5}$$

where $\mathbf{T}_{opt} = \mathbf{V}_L$ and $\mathbf{R}_{opt} = \mathbf{U}_L$ denote the optimum unconstrained precoder and combiner of this system. Here, $\mathbf{V}_L \in \mathbb{C}^{N_T\times L}$ and $\mathbf{U}_L \in \mathbb{C}^{N_R\times L}$ are L most significant right and left singular vectors of $\mathbf{H}$ [42], respectively. $\mathbf{C}_n = \mathbf{R}_{opt}^*\mathbf{R}_{opt}$ is the post-processing noise covariance matrix.

C. Hybrid BF Using Constrained SVD

In this section, we consider a mmW system shown in Figure 1. A Tx with N_T antennas and L_T RF chains communicates with a Rx with N_R antennas and L_R RF chains. We assume there are L data streams such that $L \le L_T \le N_T$ and $L \le L_R \le N_R$. At the Tx, L data streams are processed by a BB precoder $\mathbf{T}_{BB} \in \mathbb{C}^{L_T\times L}$ followed by an RF precoder $\mathbf{T}_{RF} \in \mathbb{C}^{N_T\times L_T}$. Then, the transmitted signal $\mathbf{x} \in \mathbb{C}^{N_T\times 1}$ can be written as,

$$\mathbf{x} = \mathbf{T}_{RF}\mathbf{T}_{BB}\mathbf{s}, \tag{6}$$

where $\mathbf{s} \in \mathbb{C}^{L\times 1}$ is the vector of transmitted symbols. The average total transmit power is denoted as P, and $\mathbf{s}$ satisfies $\mathbb{E}\left[\mathbf{s}\mathbf{s}^*\right] = \left(\frac{P}{L}\right)\mathbf{I}_L$. We denote the mmW channel between the Tx and Rx with $\mathbf{H} \in \mathbb{C}^{N_R\times N_T}$. The received signal over N_R antennas of the Rx is given as,

$$\mathbf{r} = \mathbf{H}\mathbf{T}_{RF}\mathbf{T}_{BB}\mathbf{s} + \mathbf{n}, \tag{7}$$

where $\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$ is the Gaussian noise vector of dimension $N_R \times 1$. Then, the Rx processes the received signal $\mathbf{r} \in \mathbb{C}^{N_R\times 1}$ with an RF combiner $\mathbf{R}_{RF} \in \mathbb{C}^{N_R\times L_R}$ followed by a BB combiner $\mathbf{R}_{BB} \in \mathbb{C}^{L_R\times L}$. The vector of received symbols $\mathbf{y} \in \mathbb{C}^{L\times 1}$ is then obtained as,

$$\mathbf{y} = \mathbf{R}_{BB}{}^*\mathbf{R}_{RF}{}^*\mathbf{H}\mathbf{T}_{RF}\mathbf{T}_{BB}\mathbf{s} + \mathbf{R}_{BB}{}^*\mathbf{R}_{RF}{}^*\mathbf{n}. \tag{8}$$

Analog and digital beamformers of a hybrid BF system need to be designed based on the constraints of power and finite-precision phase shifters, which are used in the RF domain. L_T RF-BF vectors with the dimension of $N_T \times 1$ at the Tx and L_R RF-BF vectors with the dimension of $N_R \times 1$ at the Rx are designed based on quantized directions. In particular, ith BF vector of the RF precoder and jth BF vector of the RF combiner are given as $[\mathbf{T}_{RF}]_{:,i}$, $i = 1, \ldots, L_T$ and $[\mathbf{R}_{RF}]_{:,j}$, $j = 1, \ldots, L_R$, respectively. As in the optimum BF, we can design analog and digital beamformers of a hybrid BF system by maximizing a metric (e.g., SNR, achieved rate) over all possible beamformers. By selecting the achieved rate as our metric, our goal is to design beamformers at the Tx and Rx ($\mathbf{T}_{RF}$, $\mathbf{T}_{BB}$, $\mathbf{R}_{RF}$, $\mathbf{R}_{BB}$), which maximize the rate defined in (5) while the following constraints are satisfied:

1) Due to the usage of phase shifters, the entries of $\mathbf{T}_{RF}$ and $\mathbf{R}_{RF}$ must have constant modulus. In particular, $|[\mathbf{T}_{RF}]_{i,j}|^2 = N_T^{-1}$ and $|[\mathbf{R}_{RF}]_{i,j}|^2 = N_R^{-1}$, where $|[\mathbf{T}_{RF}]_{i,j}|(|[\mathbf{R}_{RF}]_{i,j}|)$ corresponds to the magnitude of (i,j)th element of $\mathbf{T}_{RF}(\mathbf{R}_{RF})$.
2) Elements of each column in $\mathbf{T}_{RF}$ and $\mathbf{R}_{RF}$ are represented as quantized phase shifts, where each phase shifter is controlled by an N_q-bit input. $n(m)$th row of the RF precoding matrix at the Tx(Rx), which corresponds to the phase shifts of the $n(m)$th antenna of the $\mathbf{T}_{RF}(\mathbf{R}_{RF})$, can be written as $e^{\frac{j2\pi nk_q}{2^{N_q}}}\left(e^{\frac{j2\pi mk_q}{2^{N_q}}}\right)$ for some $k_q = 0, 1, \ldots, 2^{N_q} - 1$.
3) The power constraint must be satisfied, i.e., $\|\mathbf{T}_{RF}\mathbf{T}_{BB}\|_F^2 = L$ and $\|\mathbf{R}_{RF}\mathbf{R}_{BB}\|_F^2 = L$.

D. mmW Channel Model

For the scope of this paper, we consider geometric channel model. Various studies [43], [44] have shown that mmW channels have limited scattering due to the high free-space path loss. The geometric channel model, which has been proposed in [45], [46], is suitable to characterize the mathematical structure of mmW channels. In this model, each scatterer contributes a single propagation path between the Tx and the Rx. The channel representation is given as,

$$\mathbf{H} = \sqrt{\frac{N_T N_R}{\rho}} \sum_{s=1}^{S} g_s \mathbf{a}_R(\theta_s) \mathbf{a}_T^*(\phi_s), \tag{9}$$

where S is the number of scatterers, ρ is the average path-loss between the Tx and the Rx, and g_s is the complex gain of the sth path with Rayleigh distribution, i.e., $g_s \sim \mathcal{N}(0, \overline{G})$ for $s = 1, 2, \ldots, S$. Here, $\overline{G}$ denotes the average power gain. $\mathbf{a}_T(\phi_s)$ and $\mathbf{a}_R(\theta_s)$ are the array response vectors at the Tx and the Rx, respectively. $\phi_s \in [0, 2\pi]$ and $\theta_s \in [0, 2\pi]$ indicate the sth path's azimuth Angle of Arrival (AoA) and Angle of Departure (AoD), respectively. For more details of the geometric channel model we refer the reader to [45], [46].

III. DL for SVD Approximations

In this section, our objective is to leverage DL to effectively estimate the best rank-k approximation of a matrix $\mathbf{H}$, which can be defined as,

$$\mathbf{H}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k{}^*, \tag{10}$$

where $\mathbf{U}_k$ is the first k columns of left singular vectors matrix $\mathbf{U}$, $\mathbf{V}_k$ is the first k columns of right singular vectors matrix $\mathbf{V}$, and $\mathbf{\Sigma}_k$ is the diagonal matrix with top k singular values of $\mathbf{\Sigma}$ on its diagonal. The SVD provides the justified solution for a best approximation of the matrix $\mathbf{H}$ as a rank-k matrix when the error is measured in the Frobenius norm [47]. Furthermore, $\mathbf{H}_k$ can be also written as a sum of k rank-1 approximations of $\mathbf{H}$ as,

$$\mathbf{H}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i{}^*, \tag{11}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_k$ are k top singular values, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ are k top left singular vectors, and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ are k top right singular vectors of $\mathbf{H}$.

In this section, we propose three DNN architectures with different levels of complexity to meet the trade-off between complexity and accuracy. The proposed DNNs learn the best rank-k matrix approximation in a supervised manner using the factorization obtained by the SVD.

A. DNN for Rank-k Matrix Approximation

We first propose a DNN for rank-k matrix approximation, which can be seen in Figure 2. We choose CNNs to implement the proposed DNN, which can also be implemented by using different models such as feedforward, multi-layer perceptron (MLP), RNN, etc. [48]. DNN for rank-k matrix approximation learns how to predict k most significant singular values and singular vectors, i.e., $\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_k, \tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \ldots, \tilde{\mathbf{u}}_k$,

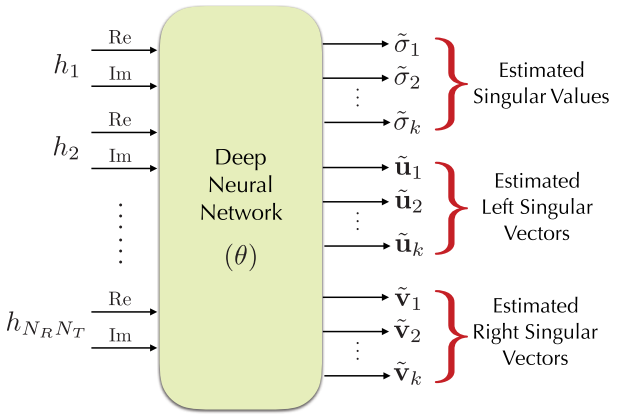


Fig. 2. DNN for rank-k matrix approximation.

and $\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \ldots, \tilde{\mathbf{v}}_k$, directly from a given matrix $\mathbf{H}$ by training its parameters θ. Consider $\mathbf{H}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i{}^*$ and $\tilde{\mathbf{H}}_k = \sum_{i=1}^{k} \tilde{\sigma}_i \tilde{\mathbf{u}}_i \tilde{\mathbf{v}}_i^*$ as real and estimated rank-k approximations of the matrix $\mathbf{H}$, respectively. The objective of the proposed DNN is to estimate the best rank-k matrix approximation of a given matrix. Therefore, we propose a custom loss function, which satisfies the following:

1) $||\mathbf{H}_k - \tilde{\mathbf{H}}_k||_F$ must be minimized.
2) $\tilde{\mathbf{U}}_k = [\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \ldots, \tilde{\mathbf{u}}_k]$ and $\tilde{\mathbf{V}}_k = [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \ldots, \tilde{\mathbf{v}}_k]$ must be unitary matrices. In particular, the columns of $\tilde{\mathbf{U}}_k$ and $\tilde{\mathbf{V}}_k$ must form a set of orthonormal vectors, which implies that $||\tilde{\mathbf{u}}_i^* \tilde{\mathbf{u}}_j||_2 = ||\tilde{\mathbf{v}}_i^* \tilde{\mathbf{v}}_j||_2 = 0 \ \forall \ i, j$ s.t. $i \neq j$.

Consequently, we define the loss function for the DNN for rank-k matrix approximation as,

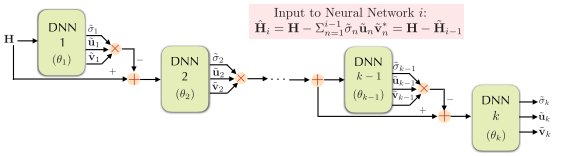
$$\mathcal{L}(\theta) = \frac{||\mathbf{H}_k - \tilde{\mathbf{H}}_k||_F}{||\mathbf{H}_k||_F} + \lambda_1 \sum_{i \neq j} ||\tilde{\mathbf{u}}_i^* \tilde{\mathbf{u}}_j||_2 + \lambda_2 \sum_{i \neq j} ||\tilde{\mathbf{v}}_i^* \tilde{\mathbf{v}}_j||_2, \tag{12}$$

where θ denotes the parameters of the DNN. Here, σ_i, $\mathbf{u}_i$, and $\mathbf{v}_i$ are the ith largest singular value and left and right singular vectors of $\mathbf{H}$, respectively. λ_1 and λ_2 are the non-negative constants of the penalty terms that satisfy $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$ to be unitary matrices.

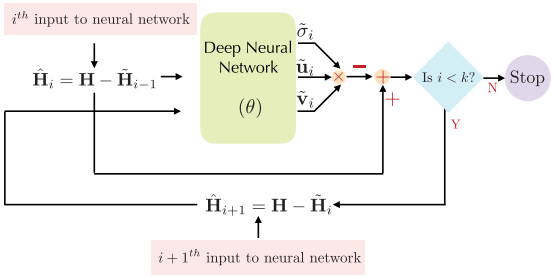
The number of output nodes increases linearly with k, N_R, and N_T in the DNN for rank-k matrix approximation. For a full-rank matrix $\mathbf{H}$, k can be as large as $\min(N_R, N_T)$, and then, the number of output nodes grows quadratically with the smaller dimension of $\mathbf{H}$.

B. Low-Complexity DNN for Rank-k Matrix Approximation

In this section, we propose a second DNN architecture, which is shown in Figure 3-a. This architecture consists of k low-complexity DNNs with the parameters denoted by θ_i, $i = 1, .., k$, in which the DNN-i is trained to estimate singular value σ_i and corresponding singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ of a given matrix $\mathbf{H}$. In other words, the DNN-i determines a function between the input matrix and its largest singular value and singular vectors by training its parameters θ_i. Given the channel matrix $\mathbf{H}$ as an input, DNN-1 generates $\tilde{\sigma}_1$, $\tilde{\mathbf{u}}_1$, and $\tilde{\mathbf{v}}_1$, which are the estimated values of σ_1, $\mathbf{u}_1$, and $\mathbf{v}_1$.



(a) Low-complexity DNN for rank-k matrix approximation.



(b) DNN for SVD via rank-1 matrix approximation.

Fig. 3. The second and third DNN architectures for the SVD, which have less complexity compared to the first DNN architecture.

We denote the input matrix for DNN-i, $i = 2, \ldots, k$ as $\hat{\mathbf{H}}_i = \mathbf{H} - \sum_{n=1}^{i-1} \tilde{\sigma}_n \tilde{\mathbf{u}}_n \tilde{\mathbf{v}}_n = \mathbf{H} - \tilde{\mathbf{H}}_{i-1}$. In particular, we represent the input of DNN-2 as $\hat{\mathbf{H}}_2 = \mathbf{H} - \tilde{\mathbf{H}}_1$, where $\tilde{\mathbf{H}}_1 = \tilde{\sigma}_1 \tilde{\mathbf{u}}_1 \tilde{\mathbf{v}}_1^*$. DNN-2 generates $\tilde{\sigma}_2$, $\tilde{\mathbf{u}}_2$, and $\tilde{\mathbf{v}}_2$. Then, $\tilde{\mathbf{H}}_2 = \sum_{i=1}^{2} \tilde{\sigma}_i \tilde{\mathbf{u}}_i \tilde{\mathbf{v}}_i^*$ is calculated, and subtracted from $\mathbf{H}$ to generate the input for DNN-3 as $\hat{\mathbf{H}}_3 = \mathbf{H} - \tilde{\mathbf{H}}_2$. This procedure continues until DNN-k gets the $\hat{\mathbf{H}}_k$ as an input and generates $\tilde{\sigma}_k$, $\tilde{\mathbf{u}}_k$, and $\tilde{\mathbf{v}}_k$.

For the training procedure of this architecture, we propose two approaches. In the first approach, k DNNs are trained jointly to minimize the total loss, which is formulated as,

$$\mathcal{L}(\theta_1, \theta_2, \ldots, \theta_k) = \frac{||\mathbf{H}_k - \tilde{\mathbf{H}}_k||_F}{||\mathbf{H}_k||_F} + \lambda_1 \sum_{i \neq j} ||\tilde{\mathbf{u}}_i^* \tilde{\mathbf{u}}_j||_2 + \lambda_2 \sum_{i \neq j} ||\tilde{\mathbf{v}}_i^* \tilde{\mathbf{v}}_j||_2, \quad (13)$$

where $\mathbf{\Theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ denotes the parameters of k DNNs to be learned. We also assume that a gradient-based technique is used to learn $\mathbf{\Theta}$. In this case, $\mathbf{\Theta}_{(t+1)}$, which corresponds to the parameters at the $(t+1)$th iteration, can be updated using the loss function given in (13) as,

$$\mathbf{\Theta}_{(t+1)} = \mathbf{\Theta}_{(t)} - \gamma \nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta})|_{\mathbf{\Theta}=\mathbf{\Theta}_t}, \quad (14)$$

where γ is the learning rate. The second approach is to train the k DNNs successively, in a sequential manner, where ith DNN is trained to learn θ_i by minimizing its own loss function. In particular, DNN-1 is trained by minimizing,

$$\mathcal{L}(\theta_1) = \frac{||\sigma_1 \mathbf{u}_1 \mathbf{v}_1^* - \tilde{\sigma}_1 \tilde{\mathbf{u}}_1 \tilde{\mathbf{v}}_1^*||_F}{||\sigma_1 \mathbf{u}_1 \mathbf{v}_1^*||_F}, \quad (15)$$

where θ_1 denotes the parameters of the first DNN in the low-complexity architecture. To satisfy $||\tilde{\mathbf{u}}_1^* \tilde{\mathbf{u}}_2||_2 = ||\tilde{\mathbf{v}}_1^* \tilde{\mathbf{v}}_2||_2 = 0$, we define the loss function of DNN-2 as,

$$\mathcal{L}(\theta_2) = \frac{||\sigma_2 \mathbf{u}_2 \mathbf{v}_2^* - \tilde{\sigma}_2 \tilde{\mathbf{u}}_2 \tilde{\mathbf{v}}_2^*||_F}{||\sigma_2 \mathbf{u}_2 \mathbf{v}_2^*||_F} + \lambda_1 ||\tilde{\mathbf{u}}_1^* \tilde{\mathbf{u}}_2||_2 + \lambda_2 ||\tilde{\mathbf{v}}_1^* \tilde{\mathbf{v}}_2||_2, \quad (16)$$

where θ_2 are the parameters of the second DNN in the low-complexity architecture. In general, the loss function of DNN-i of this architecture for successive training is defined as,

$$\mathcal{L}(\theta_i) = \frac{||\sigma_i \mathbf{u}_i \mathbf{v}_i^* - \tilde{\sigma}_i \tilde{\mathbf{u}}_i \tilde{\mathbf{v}}_i^*||_F}{||\sigma_i \mathbf{u}_i \mathbf{v}_i^*||_F} + \lambda_1 \sum_{i,j<i} ||\tilde{\mathbf{u}}_i^* \tilde{\mathbf{u}}_j||_2 + \lambda_2 \sum_{i,j<i} ||\tilde{\mathbf{v}}_i^* \tilde{\mathbf{v}}_j||_2, \quad (17)$$

where θ_i denotes the parameters of the ith DNN, λ_1 and λ_2 are non-negative constants of the penalty terms, respectively.

C. DNN for SVD via Rank-1 Matrix Approximation

For further simplicity, we propose a third DNN architecture, which predicts k singular values and singular vectors of a given matrix $\mathbf{H}$ with a single DNN recursively, as depicted in Figure 3-b. Let the matrix $\hat{\mathbf{H}}_i = \mathbf{H} - \tilde{\mathbf{H}}_{i-1}$ denote the input matrix given to the DNN in the ith iteration, where $\tilde{\mathbf{H}}_{i-1} = \sum_{n=1}^{i-1} \tilde{\sigma}_n \tilde{\mathbf{u}}_n \tilde{\mathbf{v}}_n$. Then, top singular value and singular vectors of $\hat{\mathbf{H}}_i$ are actually the ith singular value and singular vectors of $\mathbf{H}$ under the assumption that previous $i-1$ singular values and singular vectors are estimated perfectly, i.e., $\tilde{\sigma}_n = \sigma_n$, $\tilde{\mathbf{u}}_n = \mathbf{u}_n$, and $\tilde{\mathbf{v}}_n = \mathbf{v}_n$ for $n = 1, 2, \ldots, i-1$. In the first iteration, the DNN predicts $\tilde{\sigma}_1$, $\tilde{\mathbf{u}}_1$, and $\tilde{\mathbf{v}}_1$. Then, $\tilde{\mathbf{H}}_1 = \tilde{\sigma}_1 \tilde{\mathbf{u}}_1 \tilde{\mathbf{v}}_1^*$ is subtracted from the input matrix $\mathbf{H}$ to obtain $\hat{\mathbf{H}}_2 = \mathbf{H} - \tilde{\mathbf{H}}_1$. The second-highest singular value and singular vectors of $\mathbf{H}$ are estimated by providing $\hat{\mathbf{H}}_2$ to the DNN in the second iteration since top singular value and singular vectors of $\hat{\mathbf{H}}_2$ are the second-highest singular value and singular vectors of $\mathbf{H}$. This recursive procedure ends when $\tilde{\sigma}_k$, $\tilde{\mathbf{u}}_k$, and $\tilde{\mathbf{v}}_k$ are estimated by the DNN, given that $\hat{\mathbf{H}}_k = \mathbf{H} - \tilde{\mathbf{H}}_{k-1}$ as the input in the kth iteration.

This DNN architecture is trained using the following loss function,

$$\mathcal{L}(\theta) = \frac{||\mathbf{H}_k - \tilde{\mathbf{H}}_k||_F}{||\mathbf{H}_k||_F} + \lambda_1 \sum_{i \neq j} ||\tilde{\mathbf{u}}_i^* \tilde{\mathbf{u}}_j||_2 + \lambda_2 \sum_{i \neq j} ||\tilde{\mathbf{v}}_i^* \tilde{\mathbf{v}}_j||_2, \quad (18)$$

where the second and third terms are included to satisfy the orthogonality of the left and right singular vectors, i.e., $||\tilde{\mathbf{u}}_i^* \tilde{\mathbf{u}}_j||_2 = ||\tilde{\mathbf{v}}_i^* \tilde{\mathbf{v}}_j||_2 = 0$, $\forall\, i, j$ s.t. $i \neq j$. Here, θ denotes the parameters of the DNN for rank-1 approximation. At the $(t+1)$th iteration, $\theta_{(t+1)}$ are calculated as,

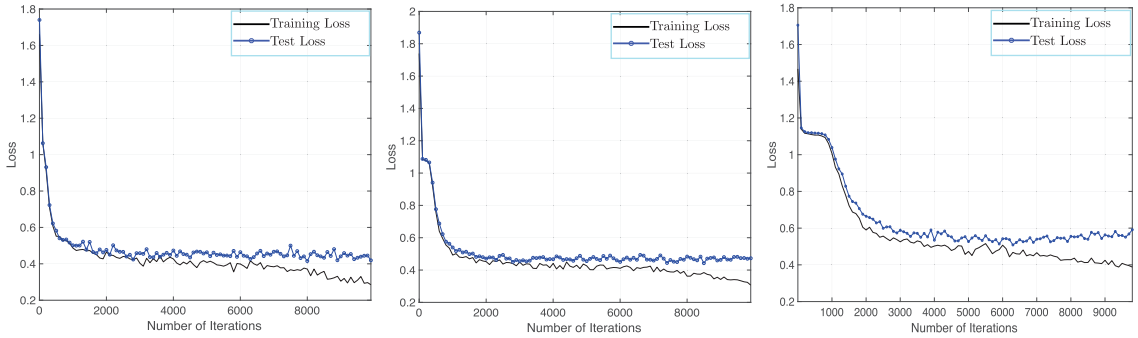
$$\theta_{(t+1)} = \theta_{(t)} - \gamma \nabla_{\theta} \mathcal{L}(\theta)|_{\theta=\theta_t}, \quad (19)$$

where γ denotes the learning rate.

D. Experimental Study of DNNs for SVD

In this section, we evaluate the performance of the proposed DNN architectures for the SVD.

1) Data Generation: We consider a dataset, which consists of 8000 training and 2000 testing channel matrices. Each of the channel matrices is generated according to the geometric channel model, as defined in (9). In this model, we assume that the spacing between two successive antennas is equal to $\lambda/2$, and we use uniform linear arrays (ULAs). We assume the AoDs/AoAs are uniformly distributed in $[0, 2\pi]$. The gain of each path in the channel has Rayleigh distribution.



(a) 8-by-8 channel matrices. (b) 16-by-16 channel matrices. (c) 32-by-32 channel matrices.

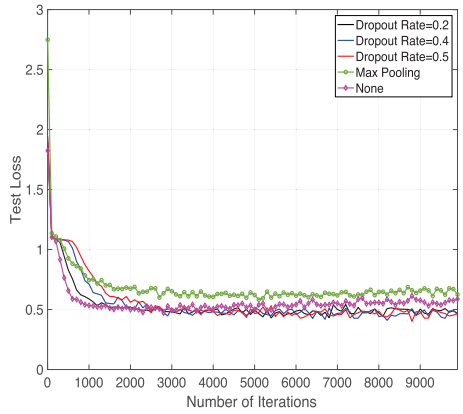
Fig. 4. Training and test losses with the DNN for rank-k matrix approximation for different sized channel matrices.

2) DL Model: Each DNN in the proposed architectures has $2N_R N_T$ inputs, which represent the real and the imaginary components of the given matrix $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$. The number of output nodes in DNN for rank-k matrix approximation equals to $k(2N_R + 2N_T + 1)$, which is the sum of k singular values $(\tilde{\sigma}_i, i = 1, 2, \ldots, k)$, and real and imaginary values of k right singular vectors $(\tilde{\mathbf{v}}_i, i = 1, 2, \ldots, k)$ and left singular vectors $(\tilde{\mathbf{u}}_i, i = 1, 2, \ldots, k)$. Here, $\tilde{\mathbf{v}}_i$ and $\tilde{\mathbf{u}}_i$ are column vectors with a size of $N_T \times 1$ and $N_R \times 1$, respectively. The number of output nodes in DNN-i of the low-complexity architecture for rank-k matrix approximation is $2N_R+2N_T+1$, which denotes the sum of $\tilde{\sigma}_i$, and real and imaginary values of $\tilde{\mathbf{v}}_i$ and $\tilde{\mathbf{u}}_i$. The DNN for SVD via rank-1 approximation also has $2N_R + 2N_T + 1$ outputs. Each DNN consists of a variable number of convolutional layers and followed by a dropout layer with a rate of 0.4 and a fully connected dense layer. Both the convolutional and the fully connected layers use exponential linear units (ELU) as activation functions [49]. The positive part of the ELU activation function has a constant gradient of one to prevent to saturate a neuron on the positive side of the function. On the other hand, it saturates exponentially on the negative side of the function, which leads to faster learning than other activation functions. We set the learning rate as 0.0001, and non-negative constants λ_1 and λ_2 for the penalty in the loss function as 0.01 unless otherwise specified. Adam [50], which is an adaptive learning rate optimization algorithm, is used for training DNNs. For the implementation, we used Tensorflow [51].

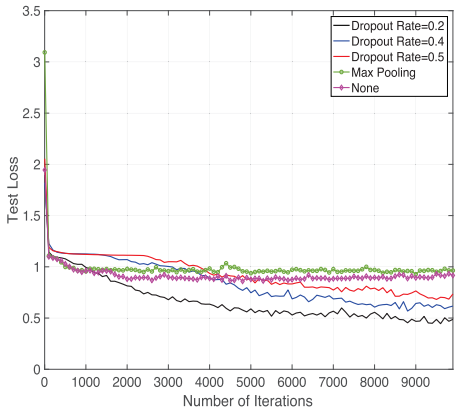
3) Comparison of Training and Test Losses: First, we obtain the training and test losses for the different-sized channel matrices using the loss function given in (12). In these simulations, we use 6 convolutional layers and a mini-batch size of 128. Figures 4-a, 4-b, and 4-c illustrate the training and test losses versus the number of iterations used during the training when $N_R = N_T = 8$, $N_R = N_T = 16$, and $N_R = N_T = 32$, respectively. The results show that the training and test losses are very close to each other for the 8-by-8 and 16-by-16 matrices while the number of iterations for the training increases up to 4000. However, test losses start to saturate, and overfitting occurs when DNNs are trained for more than 4000 iterations. Moreover, the training and test losses are nearly the same for the 32-by-32 matrices when the number of training iterations is less than 6000. These results show that reasonable test performance can be achieved by training the DNN for rank-k approximation with a higher number of training iterations while the dimension and rank of matrices increase. Therefore, overfitting starts to occur after a greater number of training iterations for the larger sized matrices. Moreover, we observe that SVD prediction error increases with the greater number of antennas at the Tx and the Rx. For instance, the SVD prediction errors obtained after the DNN is trained for 10000 iterations are 0.428, 0.494, and 0.592 for 8-by-8, 16-by-16, and 32-by-32 matrices, respectively.

4) Comparison of Dropout and Max Pooling: In this section, we study the performance of the DNN for rank-k approximation and the low-complexity DNN for rank-k approximation when the max pooling and dropout are used. Figure 5-a shows the test losses of the DNN for rank-k approximation while the number of training iterations increases up to 10000 for 16-by-16 matrices. It is seen in Figure 5-a that the test losses decrease slower with dropout compared to the case when max pooling or none of them are used. However, the smallest test losses are obtained with dropout when the DNN for rank-k approximation is trained more than 5000 iterations. In Figure 5-b, we observe the test losses versus a different number of training iterations of the low-complexity DNN for rank-k approximation using 16-by-16 matrices. For the low-complexity architecture, the smallest test losses are obtained when the dropout rate is 0.2. While the dropout rate increases up to 0.5, the performance in terms of error slightly degrades. Furthermore, smaller test losses are achieved with different rates of dropout compared to the case when max pooling or none of them are used for the higher number of training iterations. Since the dropout reduces redundancies in the DNN, it also decreases overfitting. Therefore, it outperforms max pooling in both architectures. However, the low-complexity DNN requires less generalization due to its simplicity compared to the DNN for rank-k approximation. Therefore, a lower dropout rate is required to achieve the best performance for the low-complexity DNN.

5) Impact of Selected $\mathbf{k}$ *Value During the Training and Testing:* We then investigate how the performance of the proposed DNN based SVD approach changes if the DNN is used to estimate a smaller or a larger number of singular values and singular vectors than the selected k value during the training. We first observe the performance of the proposed approach when the DNN estimates a larger number of singular

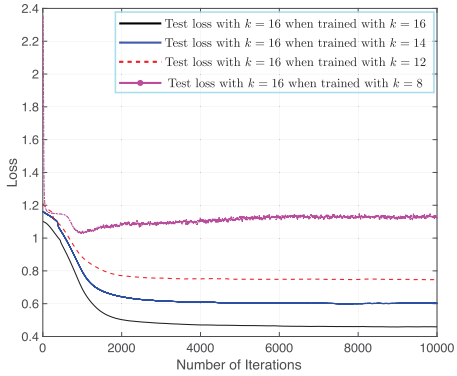


(a) DNN for rank-k approximation.

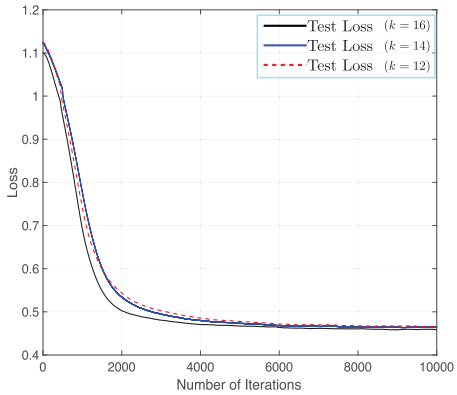


(b) Low-complexity DNN for rank-k approximation.

Fig. 5. Comparison of max pooling and dropout for 16-by-16 matrices.



(a) k is set to 16 during the testing while k changes during the training.

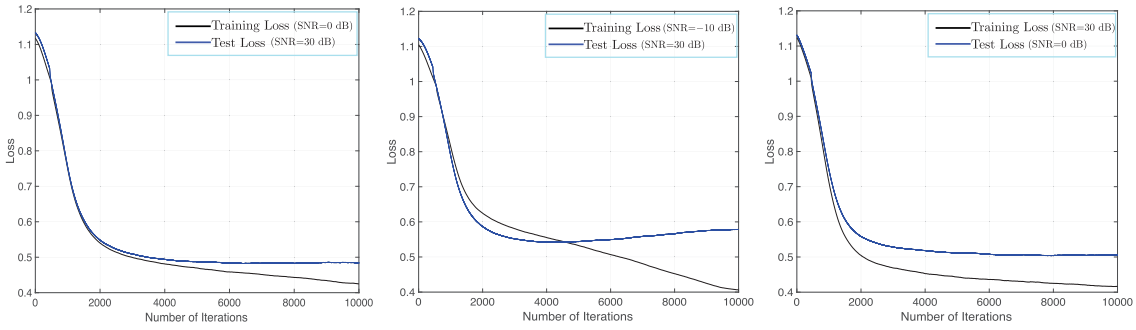


(b) k is set to 16 during the training while k changes during the testing.

Fig. 6. Test losses for 16-by-16 matrices using the DNN for rank-k matrix approximation with the different values of k during the training and testing.

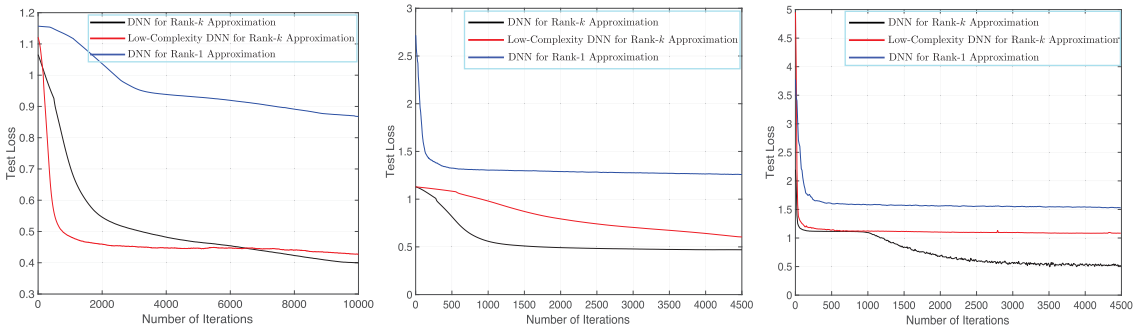
values and singular vectors than k, which is used for the training. Figure 6-a shows the test losses of 16-by-16 matrices using the DNN for rank-k matrix approximation when k is selected as 16, 14, 12, and 8 during the training. In these simulations, we use 4 convolutional layers and a mini-batch size of 32. In each case, the DNN is tested to predict 16 singular values and singular vectors of 16-by-16 matrices. It is shown in Figure 6-a that the test losses increase with the smaller k values used during the training. Therefore, the value of k in training must be at least equal to the value of k used for the testing. We then study the case when the DNN estimates a smaller number of singular values and singular vectors than the number of singular values and vectors predicted during the training. Figure 6-b illustrates the test losses of the proposed approach when the DNN estimates 16, 14, and 12 singular values and singular vectors of 16-by-16 matrices when k is set to 16 in the training phase. In this case, we observe that the testing performance of the DNN based SVD approach is not affected significantly when the DNN estimates a smaller number of singular values and vectors compared to the training phase, which implies that the DNN does not need to be retrained to estimate a smaller number of singular values and vectors than k used during the training. This result also shows that the test losses of the DNN based SVD approach do not change with the rank of the matrix when the size of the matrix remains the same.

6) Performance of the Proposed Approach in Noisy Case: In order to observe the performance of the DNN based SVD approach in noisy scenarios, we conduct the simulations using the additive white gaussian noise (AWGN) with different SNR values during the training and testing. In these simulations, we use 4 convolutional layers and a mini-batch size of 32. In Figure 7-a, we observe training and test losses of 16-by-16 channel matrices using the DNN for rank-k matrix approximation when the AWGN with 0 dB SNR and 30 dB SNR are added to the training and test matrices, respectively. As it is shown in Figure 7-a, training and test losses of the proposed approach do not degrade significantly when the noise gap between training and test matrices is 30 dB. We then increase the noise gap to 40 dB, as it is seen in Figure 7-b, where the SNR of AWGN added to the training matrices is decreased to -10 dB. As it is expected, the effect of overfitting becomes more severe due to the increased gap between the SNR used in the added noise to the training and test matrices. We then compare the case when the SNR of AWGN is higher for the training matrices compared to the SNR of AWGN added to the test matrices. In Figure 7-c, the training and test losses of 16-by-16 channel matrices are observed when the SNR of AWGN added to the training and test matrices are 30 dB and 0 dB, respectively. When the results in Figure 7-a and Figure 7-c are compared, we see that the performance degradation during the test phase due



(a) Training=0 dB, Test=30 dB. (b) Training=−10 dB, Test=30 dB. (c) Training=30 dB, Test=0 dB.

Fig. 7. Training and test losses for 16-by-16 channel matrices in the noisy case.



(a) 8-by-8 channel matrices. (b) 16-by-16 channel matrices. (c) 32-by-32 channel matrices.

Fig. 8. Comparison of the test losses for three DNNs with the different sizes of channel matrices.

to the overfitting becomes more significant in the latter case. This result occurs since the overfitting increases more if the training data is less noisy compared to the test data.

7) Comparison of the Proposed Architectures for SVD: Figures 8-a, 8-b, and 8-c illustrate the test losses with the proposed architectures for 8-by-8, 16-by-16, and 32-by-32 matrices, respectively. We use the loss functions defined in (12), (13), and (18) to train the DNN for rank-k matrix approximation, the low-complexity DNN for rank-k matrix approximation, and the DNN for SVD via rank-1 matrix approximation, respectively. We set the number of convolutional layers and the mini-batch size to 4 and 32, respectively. As shown in Figure 8-a, for 8-by-8 matrices, the low-complexity DNN for rank-k approximation outperforms the other two DNNs in terms of accuracy at the beginning of the training. However, the DNN for rank-k approximation gives smaller test losses than the other DNNs while the number of training iterations increases. We observe in Figures 8-b and Figures 8-c that the smallest test losses are obtained with the DNN for rank-k approximation when 16-by-16 and 32-by-32 matrices are used. When the matrix size is not large, the low-complexity DNN for rank-k approximation can learn faster than the DNN for rank-k approximation since the number of parameters to be learned by each DNN of the former architecture is less than the number of parameters in the latter. However, the performance of the low-complexity DNN for rank-k approximation starts to degrade with the larger sized matrices since the low-complexity of this architecture cannot deal well with more complexed data.

8) Comparison of the Proposed Training Approaches for the Low-Complexity DNN for Rank-k Approximation: We then study the performance of the two approaches proposed in Section III-B to train k DNNs of the low-complexity architecture. Figure 9 shows the test losses when the sub-DNNs in this architecture are trained jointly and sequentially for 8-by-8 matrices. In particular, we use the loss function given in (13) to train sub-DNNs of the low-complexity architecture jointly. On the other hand, DNN-i of this architecture is trained one-by-one using the loss function given in (17) in the other approach. We observe that smaller test losses are achieved when sub-DNNs are trained one-by-one. When sub-DNNs are trained sequentially, we guarantee that the first DNN is trained successfully, and the residual error occurs in the input to the next DNN decreases compared to the case when sub-DNNs are trained jointly.

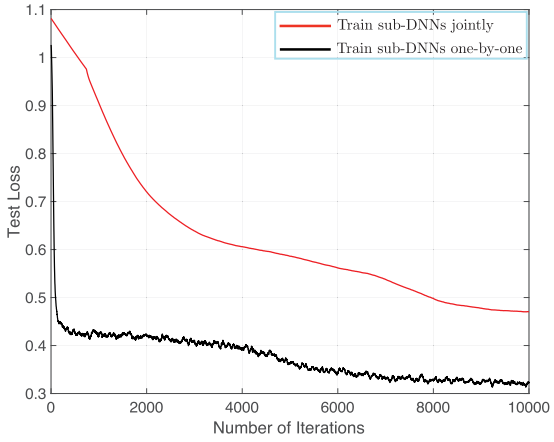
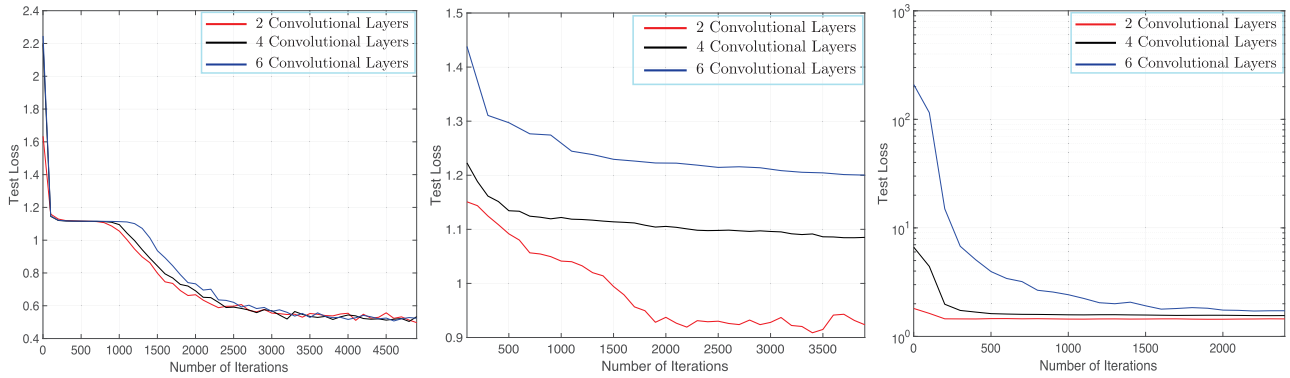
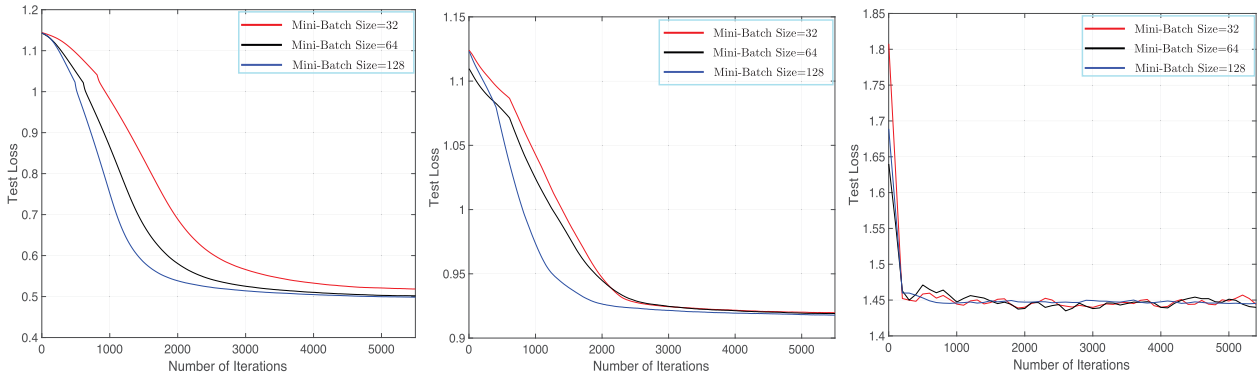


Fig. 9. Comparison of the test losses when the sub-DNNs in this architecture are trained jointly and sequentially using 8-by-8 matrices.



(a) DNN for rank-k approximation. (b) Low-complexity DNN for rank-k approximation. (c) DNN for rank-1 approximation.

Fig. 10. Test losses of the DNNs for SVD with the different number of convolutional layers using 32-by-32 channel matrices.



(a) DNN for rank-k approximation. (b) Low-complexity DNN for rank-k approximation. (c) DNN for rank-1 approximation.

Fig. 11. Test losses of the DNNs for SVD with the different sizes of mini-batches using 32-by-32 channel matrices.

9) Impact of Different Number of Convolutional Layers on the Accuracy of DNNs: In Figures 10-a, 10-b, and 10-c, we compare the test losses of 32-by-32 matrices obtained by the DNN for rank-k matrix approximation, the low-complexity DNN for rank-k approximation, and the DNN for rank-1 approximation with the different number of convolutional layers, respectively. In these results, the size of the mini-batch is set to 32. In Figure 10-a, the loss function given in (12) is used. We observe in Figure 10-a that the test losses reduce more rapidly when the number of convolutional layers is 2 and 4 in DNN for rank-k matrix approximation. While the number of iterations increases, the test losses with 6 convolutional layers become similar to the losses with 2 and 4 convolutional layers. Since the number of parameters is higher when 6 layers are used compared to 2 and 4 layers, a higher number of iterations is required for the losses to converge with 6 layers. We use the loss function given in (13) in Figure 10-b. It is shown in Figure 10-b that the smaller test losses are achieved while the number of convolutional layers decreases. Since the number of features to be learned by each DNN in the low-complexity architecture is less than the number of parameters to be learned by the DNN for rank-k approximation, the low-complexity architecture requires a smaller number of convolutional layers to achieve the maximum performance. Otherwise, the performance degrades with the higher number of convolutional layers due to overfitting. Finally, we train the DNN for rank-1 approximation using the loss function given in (18). We observe in Figure 10-c that the test losses decrease with the smaller number of convolutional layers since the DNN for rank-1 approximation also has less number of parameters to be learned compared to other proposed DNN architectures.

10) Impact of Different Sizes of Mini-Batches on the Accuracy of DNNs: In this section, we evaluate the performance of proposed DNNs in terms of accuracy with the different sizes of mini-batches. Figures 11-a, 11-b, 11-c illustrate the test losses for the DNN for rank-k matrix approximation, the low-complexity architecture, and the DNN for rank-1 approximation for 32-by-32 matrices, respectively. We use the loss function given in (12), (13), and (18) to train the DNNs. We set the number of convolutional layers to 2. The results in Figure 11-a show that the test losses decrease more rapidly while the mini-batch size grows from 32 to 128. Since the larger sizes of mini-batches provide a better estimate of the gradient [34], the test losses with a mini-batch size 64 and 128 converge to smaller values than the case with a mini-batch size 32. We observe in Figure 11-b that the test losses decrease faster with the larger sizes of mini-batches when the low-complexity DNN for rank-k approximation is used. These results reveal that the test losses converge to the global minimum more quickly with a mini-batch size of 128. In the low-complexity DNN for rank-k approximation, the global optimum is achieved eventually with the smaller sizes of

TABLE I
TIME COMPLEXITY OF CONVENTIONAL AND DNN BASED SVD METHODS

SVD Computation Technique	Time Complexity
SVD algorithm by Golub et al [19]	$O(N_R^2 N_T)$
SVD algorithm based on LAPACK routine [20]	$O(N_R N_T^2)$
DNN for rank-k approximation	$O(n \times m^2 \times f^2 \times (N_R \times N_T + p \times (N_R + N_T) + p^2) + k^2 \times (N_R^2 + N_R N_T + N_T^2))$
Low-complexity DNN for rank-k approximation	$O(k \times (n \times m^2 \times f^2 \times (N_R \times N_T + p \times (N_R + N_T) + p^2) + (N_R^2 + N_R N_T + N_T^2)))$
DNN for rank-1 approximation	$O(n \times m^2 \times f^2 \times (N_R \times N_T + p \times (N_R + N_T) + p^2) + (N_R^2 + N_R N_T + N_T^2))$

mini-batches. As shown in Figure 11-c, the test losses of the DNN for rank-1 approximation are obtained the same with the different sizes of mini-batches. Since the number of parameters of the DNN for rank-1 approximation is smaller than the other proposed DNN architectures, the test error converges rapidly with different sizes of mini-batches.

E. Comparison of Conventional Methods and DNN Based Approaches for SVD

1) Time Complexity Comparison: The conventional algorithm proposed in [19] for computing the SVD of a matrix $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$ first computes $\mathbf{H}^*\mathbf{H}$ and then calculates its eigenvalues, which gives $O(N_R^2 N_T)$ as complexity for the SVD. Another conventional method given in [20] transforms $\mathbf{H}$ into an $N_T \times N_T$ bidiagonal matrix, and then computes the singular values and singular vectors of the resulting bidiagonal matrix. The algorithm proposed in [20] has $O(N_R N_T^2)$ time complexity.

The time complexity of the training and test phases of a CNN with n convolutional layers are given as $O\left(N \times b \times \sum_{i=1}^{n} m_{i-1} \times f_i^2 \times m_i \times l_i^2\right)$ and $O\left(\sum_{i=1}^{n} m_{i-1} \times f_i^2 \times m_i \times l_i^2\right)$, respectively [52]. Here, i, m_i, f_i^2, and l_i are the index of the convolutional layer, the number of filters, the spatial size of the filter, and the output features in the ith layer, respectively. We denote the batch size of the training with b and the number of training epochs with N. Let us assume $f_i^2 = f^2$ and $m_i = m$ are fixed for $i = 1, \ldots, n$. In the proposed DNN architectures for the SVD, zero-padding with size p is applied to the input matrix with a size of $N_R \times 2N_T$. Therefore, the spatial size of output features is $l_i^2 = (N_R + 2p) \times (2N_T + 2p)$ for $i = 1, \ldots, n$. Finally, a fully-connected layer is included in each DNN architecture to generate estimated output values. The time complexity of a fully-connected layer during the testing phase can be approximated as $O\left(\sum_{i=1}^{n_{hid}} \sum_{j}^{n_{out}} d_{ij}\right)$, where n_{hid}, n_{out}, and d_i denote the number of hidden nodes, the number of output nodes, and the distance between hidden neuron i and output neuron j, respectively. The number of output nodes are $2kN_R + 2kN_T + k$ and $2N_R + 2N_T + 1$ in DNN for rank-k and rank-1 approximation, respectively. The number of output nodes of the ith DNN in the low-complexity architecture is $2N_R + 2N_T + 1$. Moreover, the number of hidden nodes in the fully-connected layer of each DNN architecture equals to number of output nodes.

Then, the time complexity of conventional SVD methods and the time complexity of the proposed DNN based approaches are obtained as in Table I. The time complexity of conventional methods can be approximated as $O(\min(N_R^2 N_T, N_R N_T^2))$. For the constant values of k, m, n, f, and p, the DNN for rank-k approximation, the low-complexity DNN, and the DNN for rank-1 approximation have a time complexity of $O(\max(k^2 N_R N_T, k^2 N_R^2, k^2 N_T^2))$, $O(\max(kN_R N_T, kN_R^2, kN_T^2))$, and $O(\max(N_R N_T, N_R^2, N_T^2))$, respectively. While the number of transmit and the number of receive antennas increase and k, m, n, f, and p are kept as constant, DNN based approaches become computationally more efficient than the conventional SVD methods. When k gets closer to N_T and N_R, the conventional methods become computationally more efficient than the DNN for rank-k approximation. The computation time of the low-complexity architecture becomes comparable to the conventional methods with the larger values of k. On the other hand, the time complexity of the DNN for rank-1 approximation is still less than the conventional methods when k approaches to N_T and N_R. We also propose to reduce the number of convolutional layers, the number of filters, and the spatial size of each filter in the low-complexity DNN for rank-k approximation. Therefore, the computational complexity can be further reduced compared to the DNN for rank-k approximation. Similarly, we propose to use a fewer number of convolutional layers and filters with a smaller spatial size compared to the DNN for rank-k approximation in the DNN for rank-1 approximation to reduce the complexity. Furthermore, the low-complexity DNN for rank-k approximation and DNN for rank-1 approximation are required to train for a fewer number of iterations, which implies that N is smaller than in the DNN for rank-k approximation. Therefore, the time complexity of the training phase is further reduced in the low-complexity DNN for rank-k approximation and DNN for rank-1 approximation.

2) Memory Requirements Comparison: The computations in each convolutional layer of a CNN require performing a convolution of each filter across the entire input. Each of the three proposed DNNs for computing the SVD gets the matrix with a size of $N_R \times 2N_T$ as the input. In each convolutional layer of the proposed DNN architecture, one filter is convolved across the input to generate an output matrix of size $(N_R + 2p) \times (2N_T + 2p)$, where p is the number of padded zeroes to each side of the input matrix. This convolution operation is repeated for each of the m filters with f^2 spatial size, producing a 1×1 strip of output values of length m. The memory requirements of the input matrix and the filter is $2N_R N_T$ and mf^2, respectively. After the entire input matrix is convolved with m filters, $2mN_R N_T + 2mpN_R + 4mpN_T + 4mp^2$ output activations are generated. The memory requirement for the output of a CNN with n convolutional layers is

TABLE II
MEMORY REQUIREMENTS FOR DNN BASED SVD METHODS

DNN for rank-k approximation	Low-complexity DNN for rank-k approximation	DNN for rank-1 approximation	k	l	m	n	t	N_R	N_T
48.9 kB	24.4 kB	2.1 kB	8	3	4	8	2	16	16
3.4 kB	2.2 kB	425.2 B	4	2	3	6	2	8	8
274.6 B	131.8 B	112.2 B	2	1	2	4	1	4	4

$2nmN_RN_T + 2nmpN_R + 4nmpN_T + 4nmp^2$. The DNN for rank-$k$ approximation is composed of multiple convolutional layers and one fully-connected layer. The fully connected layer multiplies an input vector of size $1 \times 2mN_RN_T + 2mpN_R + 4mpN_T + 4mp^2$ with a weight matrix of size $2mN_RN_T + 2mpN_R + 4mpN_T + 4mp^2 \times 2kN_R + 2kN_T + k$ to produce an output vector of size $1 \times 2kN_R + 2kN_T + k$, where $2kN_R + 2kN_T + k$ is the number of output nodes in the DNN for rank-k approximation. Therefore, the total memory requirement of the DNN for rank-k approximation equals to $k + mnf^2 + (2k+n)mp^2 + (N_R+N_T)(16kmp^2+2k) + (N_R+2N_T)(4kmp+2mnp+2mp) + N_RN_T(2+2n+4m+2km+24kmp) + kmp(N_R^2+2N_T^2) + 4k\ m(N_R^2N_T + N_RN_T^2)$.

Each of the DNN in the low-complexity architecture is composed of multiple convolutional layers and one fully-connected layer. Let us denote the number of convolutional layers and the number of filters in each DNN with $l \leq n$ and $t \leq m$, respectively. The number of output nodes of the ith DNN in this architecture is $2N_R + 2N_T + 1$ as it is explained in Section III-D. When the remaining parameters of each DNN in this architecture are the same with the DNN for rank-k approximation, the memory requirement of the fully-connected layer of each DNN is $(2N_R + 2N_T + 2)(2tN_RN_T + 2tpN_R + 4tpN_T + 4tp^2 + 1)$. Therefore, the total memory requirement of the low-complexity DNN for rank-k approximation is $k + ltf^2 + (2k+l)tp^2 + (N_R+N_T)(16ktp^2+2k) + (N_R+2N_T)(4ktp+2ltp+2tp) + N_R\,N_T(2+2l+4t+2kt+24ktp) + ktp(N_R^2+2N_T^2) + 4kt(N_R^2N_T + N_RN_T^2)$. When $l = n$ and $t = m$, the total memory requirement of the DNN for rank-k approximation and the low-complexity DNN architecture becomes equal to each other. The DNN for rank-1 approximation also consists of multiple convolutional layers and one fully-connected layer with $2N_R + 2N_T + 1$ output nodes. We assume that the values of the parameters of this architecture equal to the values of the parameters in the low-complexity DNN for rank-k approximation. The memory requirement of the fully connected layer of the DNN for rank-1 approximation is $(2N_R + 2N_T + 2)(2tN_RN_T + 2tpN_R + 4tpN_T + 4tp^2 + 1)$ as in the low-complexity architecture. Therefore, the total memory requirement of the DNN for rank-1 approximation becomes $2 + 2ltf^2 + 4tp^2(2+l) + 2tp(l+2)(N_R+2N_T) + (8tp^2+2)(N_R+N_T) + N_RN_T(2+4t+2lt+12tp) + 4t(pN_R^2 + 2pN_T^2 + N_R^2N_T)$. This result shows that a $1/k$ reduction in the memory requirement is obtained with the DNN for rank-1 approximation compared to the DNN for rank-k approximation and the low-complexity DNN architecture. We assume that each element of the real-valued arrays in the proposed architectures is represented with a floating-point number, which is stored using four bytes (32-bits). When $f^2 = 9$ and $p = 2$, the total memory requirements of the proposed DNN architectures are obtained in terms of bytes (B) and kilobytes (kB) for different values of the parameters as in Table II.

IV. DL FOR HYBRID BF

In this section, we present a novel DL-based approach for the hybrid BF system, as depicted in Figure 1. The problem of the hybrid BF system design is formulated as,

$$\begin{aligned} \left(\mathbf{T}_{RF}^{opt}, \mathbf{T}_{BB}^{opt}, \mathbf{R}_{RF}^{opt}, \mathbf{R}_{BB}^{opt}\right) &= \underset{\mathbf{T}_{RF},\mathbf{T}_{BB},\mathbf{R}_{RF},\mathbf{R}_{BB}}{\text{maximize}} R, \\ \text{s.t.} \quad \|\mathbf{T}_{RF}\mathbf{T}_{BB}\|_F^2 &= L, \quad \|\mathbf{R}_{RF}\mathbf{R}_{BB}\|_F^2 = L, \end{aligned} \tag{20}$$

where R can be obtained by substituting $\mathbf{T}_{opt} = \mathbf{T}_{RF}\mathbf{T}_{BB}$ and $\mathbf{R}_{opt} = \mathbf{R}_{RF}\mathbf{R}_{BB}$ in (5).

To solve this problem, we introduce a novel DNN based hybrid BF approach, which can be realized by using either of the three architectures proposed for the SVD in Section III. In the proposed approach, we minimize the Frobenius distance between the rank-k approximations obtained with the unconstrained and hybrid beamformers instead of maximizing the rate directly. Figure 12 depicts the proposed architecture for the hybrid BF in which the DNN for rank-k matrix approximation is used. The DNN gets the channel matrix $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$ as the input and transforms that into a real-valued matrix with $N_R \times 2N_T$ size. As shown in Figure 12, the DNN for rank-k matrix approximation, which consists of multiple convolutional layers and one fully-connected layer, is trained to estimate the L largest singular values $(\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_L)$, the unnormalized values of the BB precoder $\left([\hat{t}_1^{BB}, \ldots, \hat{t}_{L_TL}^{BB}]^T\right)$, the unnormalized values of the BB combiner $\left([\hat{r}_1^{BB}, \ldots, \hat{r}_{L_RL}^{BB}]^T\right)$, the unquantized values of the RF precoder $\left([\hat{t}_1^{RF}, \ldots, \hat{t}_{N_TL_T}^{RF}]^T\right)$, and the unquantized values of the RF combiner $\left([\hat{r}_1^{RF}, \ldots, \hat{t}_{N_RL_R}^{RF}]^T\right)$. Here, L denotes the number of data streams sent through the hybrid BF system. Then, the L largest singular values are transformed into a diagonal matrix $\tilde{\mathbf{\Sigma}}_L$ with $\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_L$ on its diagonal. Through the quantization layers, the phase value of each unquantized element of the RF precoder and combiner is quantized, and the quantized elements of the RF precoder and combiner are estimated as $[\tilde{t}_1^{RF}, \ldots, \tilde{t}_{N_TL_T}^{RF}]^T$ and $[\tilde{r}_1^{RF}, \ldots, \tilde{t}_{N_RL_R}^{RF}]^T$, respectively. The quantized elements of the RF precoder and combiner are transformed into the RF precoder and combiner matrices as $\tilde{\mathbf{T}}_{RF}$ and $\tilde{\mathbf{R}}_{RF}$, respectively. The unnormalized values of the BB precoder $\left([\hat{t}_1^{BB}, \ldots, \hat{t}_{L_TL}^{BB}]^T\right)$ and the unnormalized values of the BB combiner $\left([\hat{r}_1^{BB}, \ldots, \hat{r}_{L_RL}^{BB}]^T\right)$) are also turned into the unnormalized BB precoder and combiner as $\hat{\mathbf{T}}_{BB}$ and $\hat{\mathbf{R}}_{BB}$, respectively. Finally, the normalized BB precoder $\left(\tilde{\mathbf{T}}_{BB}\right)$ and the normalized BB combiner $\left(\tilde{\mathbf{R}}_{BB}\right)$ are estimated by using the normalization layers.

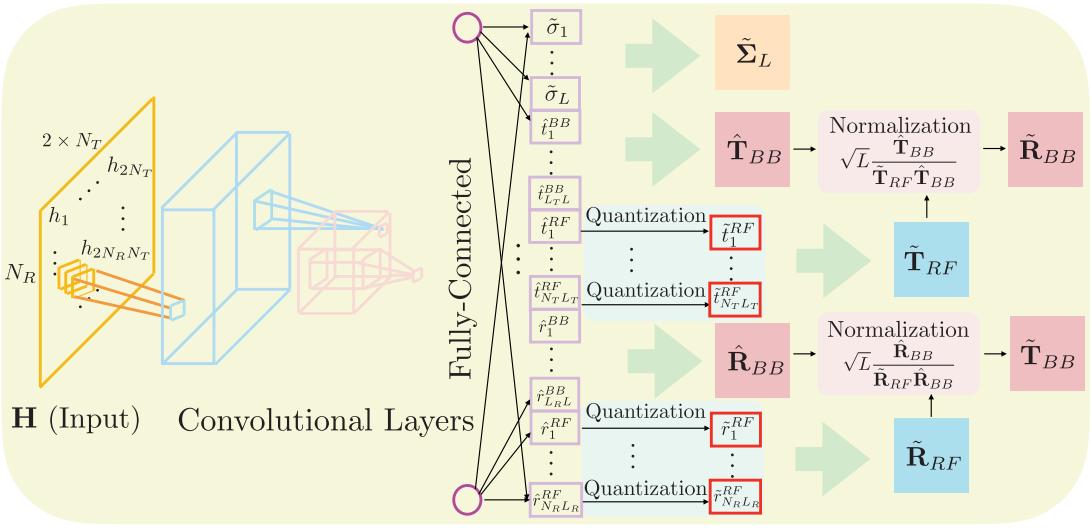


Fig. 12. DL-based hybrid BF architecture, which uses the DNN for rank-k matrix approximation.

In this section, we explain the RF constraints that we incorporate into our DNN based hybrid BF approach. Then, we describe how we satisfy the power constraint and explain the loss function used during the optimization step. Finally, we summarize our experimental results.

A. Incorporation of RF Constraints

As the finite-precision phase shifters are used in the RF domain, the elements of $\mathbf{T}_{RF} \in \mathbb{C}^{N_T \times L_T}$ and $\mathbf{R}_{RF} \in \mathbb{C}^{N_R \times L_R}$ are restricted to satisfy $|[\mathbf{T}_{RF}]_{n,i}|^2 = N_T^{-1}$ and $|[\mathbf{R}_{RF}]_{m,j}|^2 = N_R^{-1}$, respectively. If each phase shifter in the analog beamformers is controlled by an N_q-bit input, $n(m)$th row of the $[\mathbf{T}_{RF}]_{(n,i)}$ $\left([\mathbf{R}_{RF}]_{(m,j)}\right)$ is denoted by $e^{\frac{j2\pi nk_q}{2^{N_q}}}\left(e^{\frac{j2\pi mk_q}{2^{N_q}}}\right)$ for some $k_q = 0, 1, \ldots, 2^{N_q} - 1$. To incorporate these constraints, we add quantization layers to quantize the phase of each element of the RF beamformers in this architecture. A naive approach would be discretizing the weights associated with the RF beamformers using a uniform quantizer, in which each weight is rounded to the nearest value from a finite set of quantization levels. However, gradient-based optimization techniques would generate zero gradients during the training of the uniform quantizer, which would prevent to update the weights associated with the quantization layers. We propose four approaches to formulate the quantization as a differentiable function. Let us denote the ith element of the unquantized and vectorized RF precoder estimated by the DNN as $\hat{t}_i^{RF} = c_t e^{j\alpha_i}$, where $c_t = \frac{1}{\sqrt{N_T}}$ is the modulus and α_i is the phase of the ith element. Similarly, we can define the kth element of the unquantized and vectorized RF combiner as $\hat{r}_k^{RF} = c_r e^{j\beta_k}$ where $c_r = \frac{1}{\sqrt{N_R}}$ is the modulus and β_k is the phase of kth element. This section explains how the proposed approaches can be used to estimate the quantized RF precoder $\tilde{\mathbf{T}}_{RF}$. The same procedures can be applied to estimate the quantized RF combiner $\tilde{\mathbf{R}}_{RF}$, which is omitted due to the page limit.

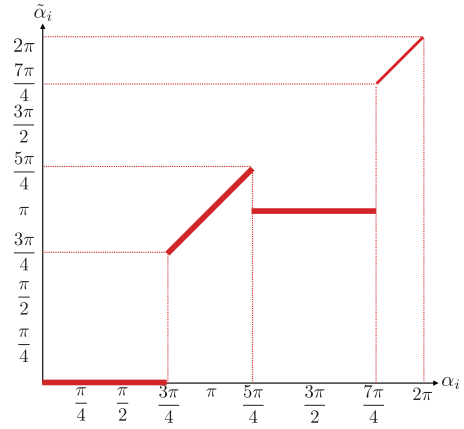
1) Quantization Approach **1***:* In this approach, we use a combination of step and piece-wise linear functions to approximate uniform quantization. Such a quantization function has non-zero gradients on the regions that are determined by piece-wise linear functions so that it can be learned during backpropagation. In this case, the weights related to the quantized RF precoder and combiner are updated in each training iteration, which would not be possible with the uniform quantization. In the training, the phase of ith element $\hat{t}_i^{RF}$ of the unquantized and vectorized RF precoder $[\hat{t}_1^{RF}, \ldots, \hat{t}_{N_T L_T}^{RF}]^T$ is approximated as,

$$\tilde{\alpha}_i = \begin{cases} 0, & \text{if } 0 \le \alpha_i \le \frac{2\pi(n-\gamma)}{2^{N_q}} \\ \alpha_i, & \text{if } \frac{2\pi(n-\gamma)}{2^{N_q}} < \alpha_i \le \frac{2\pi(n+\gamma)}{2^{N_q}}, \\ & n = 1, \ldots, 2^{N_q} - 1 \\ \frac{2\pi n}{2^{N_q}}, & \text{if } \frac{2\pi(n+\gamma)}{2^{N_q}} < \alpha_i \le \frac{2\pi((n+1)-\gamma)}{2^{N_q}}, \\ & n = 1, \ldots, 2^{N_q} - 1 \\ \alpha_i, & \text{if } \frac{2\pi(2^{N_q}-\gamma)}{2^{N_q}} < \alpha_i \le 2\pi \end{cases} \tag{21}$$

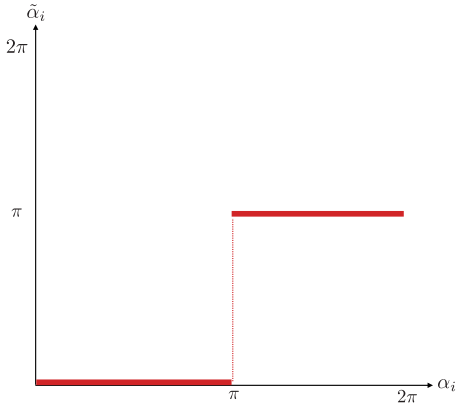
where $0 \le \gamma \le 1$. $\tilde{\alpha}_i$ is the quantized phase value based on the piece-wise linear approximations. Then, the quantized value of the ith element in the RF precoder can be written as $\tilde{t}_i^{RF} = c_t e^{j\tilde{\alpha}_i}$ for $1 \le i \le N_T L_T$. The quantized and vectorized RF precoder $[\tilde{t}_1^{RF}, \ldots, \tilde{t}_{N_T L_T}^{RF}]^T$ is then reshaped into the quantized RF precoder matrix given as $\tilde{\mathbf{T}}_{RF}$. In Figures 13-a and 13-b, toy examples for this quantization technique are presented. In Figure 13-a, γ and N_q are set to 0.25 and 1, respectively. Figure 13-b shows the case when $\gamma = 0$ and $N_q = 1$. In the testing phase, α_i is quantized as,

$$\tilde{\alpha}_i = \frac{2\pi n}{2^{N_q}}, \tag{22}$$

where $\frac{2\pi n}{2^{N_q}} \le \alpha_i \le \frac{2\pi(n+1)}{2^{N_q}}$ and $n = 0, 1, \ldots, 2^{N_q} - 1$. As depicted in Figures 13-a and 13-b, quantization approach 1 starts to behave as the uniform quantization while γ goes to 0. When $\gamma = 0$, quantization is realized in an exactly same way in training and test phases. On the other hand, the differentiable regions get larger while γ gets closer to 1,



(a) $\gamma = 0.25$ and $N_q = 1$.



(b) $\gamma = 0$ and $N_q = 1$.

Fig. 13. Toy examples for quantization approach 1 based on piece-wise linear functions.

which would allow to update the weights related to the RF precoder and combiner in every iteration during the training.

2) Quantization Approach **2***:* The first proposed quantization function is not smooth and has zero gradients on the regions defined by step functions. Therefore, we replace each step function in the uniform quantization with a sigmoid function in the second quantization approach. The sigmoid function has non-zero gradients everywhere, which prevents gradient mismatch during backpropagation. For a set of phase values of the unquantized and vectorized RF precoder of a hybrid BF system with N_T transmit antennas and L_T RF chains at the Tx ($\{\alpha_i, i = 1, \ldots, N_T L_T\}$), the second quantization approach is applied to each α_i as,

$$\tilde{\alpha}_i = \frac{1}{1 + \exp\left(\beta(\alpha_i - b_n)\right)} + o_n, \tag{23}$$

where β is the scale factor of the input. b_n and o_n are the bias and offset for the nth quantization level, respectively. Here, $n = 1, \ldots, 2^{N_q}$ and N_q is the number of bits used in phase shifters. A toy example for quantization approach 2 is shown in Figure 14. In the toy example, $N_q = 1$, $b_1 = 1$, $b_2 = 2$, $o_1 = 0$, $o_2 = 1$, and $\beta = -20$. This approach converges to the uniform quantization as the absolute value of β increases.

3) Quantization Approach **3***:* The main idea of all proposed quantization approaches is to approximate the quantization operation as a differentiable function to update any weights and activations during backpropagation in DNNs. Since the weights are not updated during forward propagation, we use step functions to apply uniform quantization in the third approach for forward propagation. For N_q-bit phase shifters, uniform quantization considers $2^{N_q} - 1$ equally spaced points between 0 and 2π excluding endpoints and assigns the phase value of the ith element of the unquantized and vectorized RF precoder (α_i) to the closest quantization value. During backpropagation, we use a linear combination of sigmoid functions. In particular, the quantized value of the ith element in the RF precoder, which is written as $\tilde{t}_i^{RF} = c_t e^{j\tilde{\alpha}_i}$ for $1 \leq i \leq N_T L_T$, is computed as in (23). Here, N_T and L_T denote the number of transmit antennas and the number of RF chains at the Tx, respectively. The gap between the step and sigmoid functions can be reduced by increasing the absolute value of the scale factor β in (23). In order to visualize the difference between forward propagation and backward propagation, we present toy examples as given in Figures 15-a and 15-b. In both of the figures, $N_q = 1$. During backpropagation as shown in Figure 15-b, we set the values of b_1, b_2, o_1, and o_2 to 1, 2, 0, and 1, respectively.

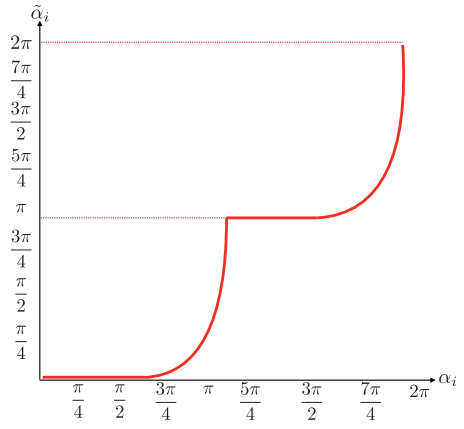
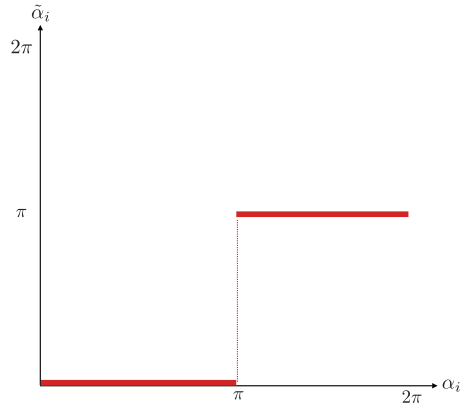


Fig. 14. Toy example for quantization approach 2 based on sigmoid functions.

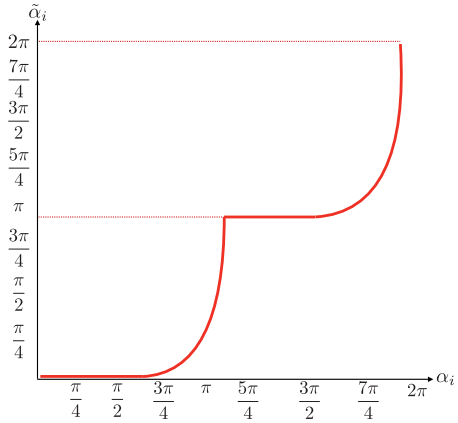
4) Quantization Approach **4***:* Finally, we propose a fourth quantization approach, which assigns α_i to one of 2^{N_q} quantization points probabilistically during forward propagation. Here, α_i denotes the phase value of the ith element in the unquantized and vectorized RF precoder, i.e., $\hat{t}_i^{RF} = c_t e^{j\alpha_i}$ for $1 \leq i \leq N_T L_T$. We apply the stochastic quantization approach given in [37] to each $\{\alpha_i, i = 1, \ldots, N_T L_T\}$ as,

$$\tilde{\alpha}_i = \frac{\lfloor 2^{N_q} \alpha_i \rfloor}{2^{N_q}} + \frac{r_i}{2^{N_q}}, \tag{24}$$

where r_i is the rounding function and defined as $r_i \sim Bernoulli\left(2^{N_q}\alpha_i - \lfloor 2^{N_q}\alpha_i \rfloor\right)$. N_q, N_T, and L_T denote the number of bits used in phase shifters, the number of transmit antennas, and the number of RF chains at the Tx, respectively. To backpropagate the gradients through this quantization function, we use the straight-through estimator as defined in [38]. Let us denote the quantization function given in (24) as $Q(\alpha)_i = \frac{\lfloor 2^{N_q}\alpha_i \rfloor}{2^{N_q}} + \frac{r_i}{2^{N_q}}$. Then, the gradient of $Q(\alpha)_i$ with respect to α_j is defined almost everywhere, and



(a) Forward propagation.



(b) Backpropagation.

Fig. 15. Toy examples for quantization approach 3 during forward and backward propagation.

it is given as,

$$\frac{\partial Q(\alpha)_i}{\tilde{\alpha}_j} = \begin{cases} 1, & \text{if } \alpha_i \text{ has been quantized to } \tilde{\alpha}_j \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

Therefore, all the weights, whose gradients are generated using the straight-through estimator, can be updated during backpropagation.

B. Satisfying Power Constraints

We also require to design the analog and digital beamformers by considering the power constraints. As we defined in Section II-C, the hybrid BF system must satisfy the power constraint, i.e., $\|\mathbf{T}_{RF}\mathbf{T}_{BB}\|_F^2 = L$ and $\|\mathbf{R}_{RF}\mathbf{R}_{BB}\|_F^2 = L$, where L is the number of transmitted data streams. To meet with the power constraints of the hybrid beamformers, we append normalization layers to the DNN, which normalize the vectorized and unnormalized values of the BB precoder and combiner generated by the DNN. Let denote the vectorized and unnormalized BB precoder with $[\hat{t}_1^{BB}, \ldots, \hat{t}_{L_T L}^{BB}]^T$ and the vectorized and unnormalized BB combiner with $[\hat{r}_1^{BB}, \ldots, \hat{r}_{L_R L}^{BB}]^T$. Then, $[\hat{t}_1^{BB}, \ldots, \hat{t}_{L_T L}^{BB}]^T$ and $[\hat{r}_1^{BB}, \ldots, \hat{r}_{L_R L}^{BB}]^T$ are transformed into the unnormalized BB precoder matrix $\hat{\mathbf{T}}_{BB}$ and the unnormalized BB combiner matrix $\hat{\mathbf{R}}_{BB}$. By using the unnormalized BB precoder $\hat{\mathbf{T}}_{BB}$, the quantized RF precoder $\tilde{\mathbf{T}}_{RF}$, the unnormalized BB combiner $\hat{\mathbf{R}}_{BB}$, and the quantized RF combiner $\tilde{\mathbf{R}}_{RF}$, the normalized BB precoder and combiner are computed as,

$$\tilde{\mathbf{T}}_{BB} = \sqrt{L}\frac{\hat{\mathbf{T}}_{BB}}{||\tilde{\mathbf{T}}_{RF}\hat{\mathbf{T}}_{BB}||_F}, \tag{26}$$

$$\tilde{\mathbf{R}}_{BB} = \sqrt{L}\frac{\hat{\mathbf{R}}_{BB}}{||\tilde{\mathbf{R}}_{RF}\hat{\mathbf{R}}_{BB}||_F}. \tag{27}$$

C. Overall Loss Function

In this subsection, we define a customized overall loss function to train the DNN for the hybrid BF. Let $\mathbf{\Sigma}_L = \text{diag}(\sigma_1, \ldots, \sigma_L)$, $\mathbf{U}_L = [\mathbf{u}_1, \ldots, \mathbf{u}_L]$, and $\mathbf{V}_L = [\mathbf{v}_1, \ldots, \mathbf{v}_L]$ denote the L largest singular values and singular vectors, where $L \leq \text{rank}(\mathbf{H})$ is the number of data streams in the hybrid BF system. We can define the rank-L matrix approximation of $\mathbf{H}$ as,

$$\mathbf{H}_L = \mathbf{U}_L\mathbf{\Sigma}_L{\mathbf{V}_L}^*. \tag{28}$$

By using the outputs of DNN, we approximate the left and right singular vectors of $\mathbf{H}$ as $\tilde{\mathbf{R}}_{opt} = \tilde{\mathbf{R}}_{RF}\tilde{\mathbf{R}}_{BB}$ and $\tilde{\mathbf{T}}_{opt} = \tilde{\mathbf{T}}_{RF}\tilde{\mathbf{T}}_{BB}$, respectively. Then, rank-L approximation of $\mathbf{H}$ is estimated as,

$$\tilde{\mathbf{H}}_L = \tilde{\mathbf{R}}_{opt}\tilde{\mathbf{\Sigma}}_L\tilde{\mathbf{T}}_{opt}^*. \tag{29}$$

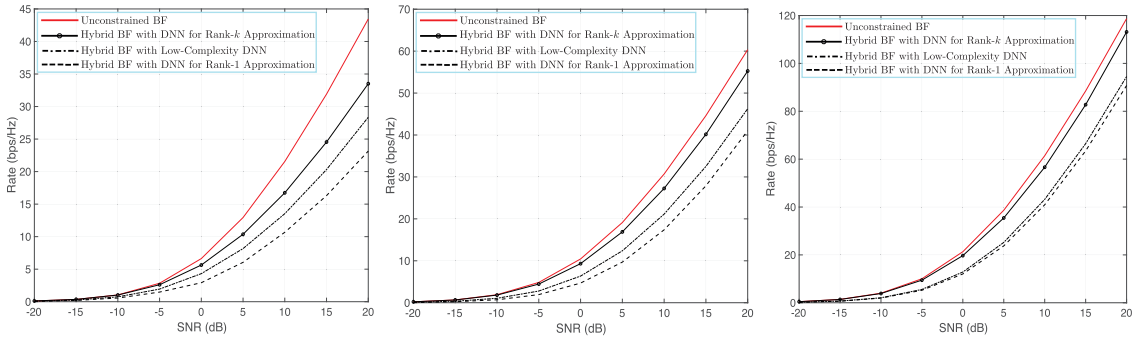
The DNN for the hybrid BF is trained to minimize the Frobenius distance between $\mathbf{H}_L$ and $\tilde{\mathbf{H}}_L$. Additionally, $\tilde{\mathbf{T}}_{opt} = [\tilde{\mathbf{t}}_1, \ldots, \tilde{\mathbf{t}}_L]$ and $\tilde{\mathbf{R}}_{opt} = [\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_L]$ must be orthogonal matrices, i.e., $||\tilde{\mathbf{t}}_i^*\tilde{\mathbf{t}}_j||_2 = ||\tilde{\mathbf{r}}_i^*\tilde{\mathbf{r}}_j||_2 = 0 \ \forall \ i, j$ s.t. $i \neq j$. Here, $\tilde{\mathbf{t}}_i \in \mathbb{C}^{N_T \times 1}$, $\tilde{\mathbf{r}}_i \in \mathbb{C}^{N_R \times 1}$, and $i = 1, \ldots, L$. Formally, we define the loss function as,

$$\mathcal{L}(\theta) = \frac{||\mathbf{H}_L - \tilde{\mathbf{H}}_L||_F}{||\mathbf{H}_L||_F} + \lambda_1 \sum_{i \neq j} ||\tilde{\mathbf{r}}_i^*\tilde{\mathbf{r}}_j||_2 + \lambda_2 \sum_{i \neq j} ||\tilde{\mathbf{t}}_i^*\tilde{\mathbf{t}}_j||_2, \tag{30}$$

where λ_1 and λ_2 are the non-negative constants. The second and third terms in (30) satisfy the orthogonality of the $\tilde{\mathbf{T}}_{opt}$ and $\tilde{\mathbf{R}}_{opt}$, i.e., $||\tilde{\mathbf{r}}_i^*\tilde{\mathbf{r}}_j||_2 = 0$ and $||\tilde{\mathbf{t}}_i^*\tilde{\mathbf{t}}_j||_2 = 0$, $\forall \ i, j$ s.t. $i \neq j$.

D. Experimental Study of DNNs for Hybrid BF

In this subsection, we compare the achieved rates with the proposed hybrid BF approach based on three DNNs for the SVD. We highlight the impact of different system parameters, such as the number of antennas, the number of iterations, and the types of quantization techniques, on the performance of our approach. Moreover, we compare the achieved rates of the proposed hybrid BF based on three DNNs with the unconstrained BF, the conventional hybrid BF algorithms given in [9], [10], [39], an ML-aided hybrid BF algorithm based on adaptive CE optimization [25], two DL-based hybrid BF algorithms [27], [30], an autoencoder based hybrid BF algorithm [28]. For the simulations of [9], the parameters are chosen as follows. The number of BF vectors at the Tx in each stage of the algorithm is set to 2. The required resolutions for the AoD and AoA are chosen as $2N_T$ and $2N_R$, respectively. In the simulations of [10], we set the number of paths of



(a) 8-by-8 mmW system. (b) 16-by-16 mmW system. (c) 32-by-32 mmW system.

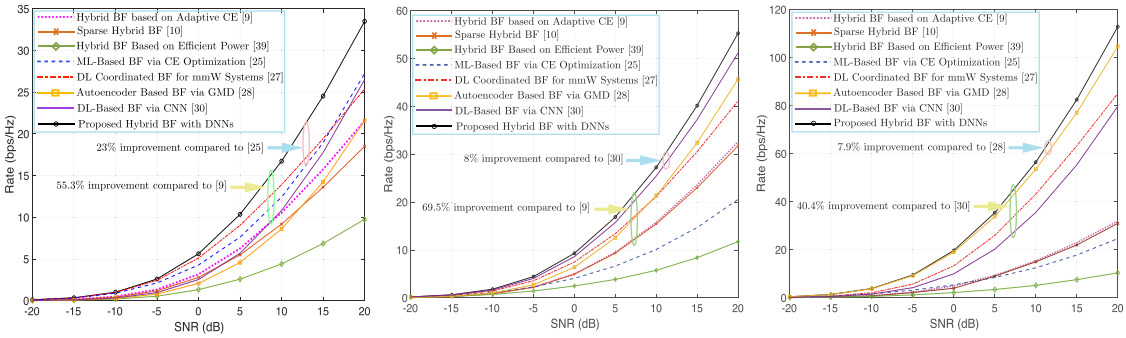
Fig. 16. Achievable rates of the proposed approach for hybrid BF based on three DNNs for SVD, and the unconstrained BF versus SNR for different number of Tx and Rx antennas.

the channel to the $\max\{N_T, N_R\}$ unless otherwise specified. The simulation parameters of the algorithm proposed in [39] are chosen as follows. Q_t and Q_r, which denote the Tx and Rx beams with the largest effective powers selected for the reduced set of the possible beams, are assumed to equal to each other, i.e., $Q_t = Q_r = Q = 2^{N_q}$. Here, N_q is the number of bits used in the phase shifters. For the simulations of [25], the number of candidates and elites for the beamformers are set to 2×2^{N_q} and 2^{N_q}, respectively. We use a step size of 0.0001, and train the algorithm for 1000 iterations. In the simulations of [27], we choose the parameters as follows. For the training of the algorithm, an MLP with 4 hidden layers is used. The ReLU activation functions are used in the hidden layers. The dropout rate and the learning rare are set to 0.05 and 0.0001, respectively. We use a mini-batch size of 100. The MLP is trained using the mean squared error function with Adam optimization. The best weights are found using early stopping to avoid overtraining. 10000 channel matrices are divided into a training set with 8000 matrices and a test set with 2000 matrices. The number of BS and the mobile user is set to 1, and the number of antennas at the BS and the user are selected as equal to each other, i.e., $N_T = N_R$. For [30], a CNN with four convolutional layers is used in the simulations. The number of filters in the first, second, and third convolutional layers in the CNN are 32, 64, and 128, respectively. The last convolutional layer has 128 filters. The spatial size of each filter in the CNN is set to 9, and the size of zero-padding is 7. The CNN is trained using the mean squared error function with Adam optimization. The learning rate used during the training is 0.0001. Finally, the dropout rate, the mini-batch size, the number of epochs are set to 0.05, 32 and 1000, respectively. In [28], the number of training iterations, the learning rate, and the mini-batch size are selected as 1000, 0.0001, and 32, respectively. For the simulations of all the algorithms, the number of bits used in the phase shifters is set to 2 for 8-by-8 mmW systems, and 3-bit phase shifters are used in the 16-by-16 and 32-by-32 mmW systems. Moreover, the number of paths in the channel equals the $\max\{N_T, N_R\}$ unless otherwise specified.

1) DL Model: We design the DNNs in our hybrid BF approach using CNNs, as in Section III-D. We use mini-batches with a size of 32, and we obtain the simulation results for DNN with 4 convolutional layers. The first convolutional layer uses a filter with a size of $32 \times 3 \times 3$. The second and third convolutional layers apply filters with a size of $64 \times 3 \times 3$. In the remaining layers, filters with a size of $128 \times 3 \times 3$ are used. Convolutional layers are followed by a dropout layer with a rate of 0.4, a max pooling layer, and a fully connected dense layer. DNNs for the hybrid BF have ELU activation units in each hidden layer except the quantization layers, which use sigmoid functions instead. Finally, the linear activation function is used in the last layer. The quantization approach 1 based on piece-wise linear functions is used to generate the phases of RF beamformers during the training of each DNN unless otherwise specified. The loss function given in (30) with Adam optimization is used for training DNNs. The test and training losses of the algorithm with different sized matrices are observed while changing the values of non-negative constants from 0.0001 to 0.1 with the step of 0.001. Since the best performance is achieved when both of the non-negative constants equal to 0.01, λ_1 and λ_2 are set to 0.01. The rates are obtained after training DNNs for 1000 iterations with a 0.0001 learning rate unless otherwise specified. We use the Keras [53] with a Tensorflow [51] backend for the simulations.

2) Channel Model and Data Generation: For generating datasets to represent the mmW channel in different time instances, we use the geometric channel model introduced in Section II. In the simulations of the geometric channel model, we assume the antenna arrays to be ULAs. The spacing between two successive antennas is equal to $\lambda/2$. The AoDs/AoAs are uniformly distributed in $[0, 2\pi]$. The distance between the Tx and Rx is 50 meters (m), the carrier frequency is 28 GHz, the path loss exponent (PLE) is 3, and the system bandwidth is 100 MHz. The average transmit power is set to 7 dB. 10000 channel matrices, which are divided into a training set with 8000 matrices and a test set with 2000 matrices, are generated for 8-by-8, 16-by-16, 32-by-32, and 64-by-64 mmW systems.

3) Comparison of Hybrid BF Approaches Based on Proposed DNNs for SVD: To study the performance of the proposed hybrid BF based on DNNs, we first conduct simulations using the geometric channel model given in Section II-D. We implement each DNN using a mini-batch size of 32. The number of convolutional layers is set to 4. We generate



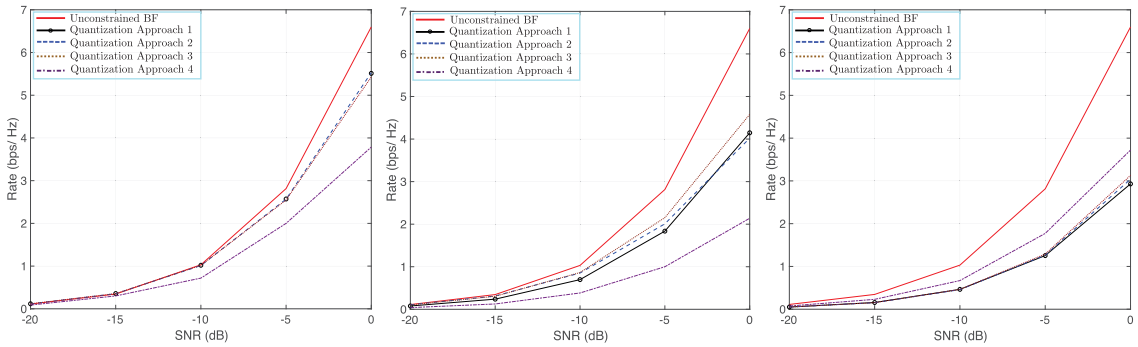
(a) 8-by-8 mmW system. (b) 16-by-16 mmW system. (c) 32-by-32 mmW system.

Fig. 17. Achieved rates of DNN based hybrid BF, conventional hybrid BF algorithms given in [9], [10], [39], and the data-driven hybrid BF algorithms given in [25], [27], [28], [30] for mmW systems with different number of Tx and Rx antennas.

full-rank channel matrices for all three cases. Therefore, the number of channel paths (S) equals the number of transmit and receive antennas of the mmW system. Figures 16-a, 16-b, and 16-c illustrate the achieved rates with the proposed hybrid BF approach based on three DNNs for the SVD, and the unconstrained BF for 8-by-8 mmW system with 2-bit phase shifters, 16-by-16 and 32-by-32 mmW systems with 3-bit phase shifters. For different sized mmW systems, the hybrid BF approach based on DNN for rank-k matrix approximation outperforms the hybrid BF based on the low-complexity DNN for rank-k approximation and the hybrid BF based on the DNN for rank-1 approximation. In these results, we observe that the performance gap between the DNN for rank-k matrix approximation and the other DNN architectures increases as the number of antennas increases at the Tx and the Rx. The reason of this behavior is that an additional distortion is added to the input matrix when ith singular value and singular vectors are estimated from rank-$k{-}i{+}1$ matrix obtained by subtracting rank-$i-1$ matrix, which is computed using the estimated $i-1$ singular values and vectors, from the original rank-k channel matrix. Therefore, the distortion increases as the rank of the channel increases for the low-complexity DNN for rank-k matrix approximation and the DNN for rank-1 matrix approximation. Since we consider full-rank channel matrices in these simulations, the number of transmit and receive antennas is equal to the rank of the channel, which leads to an increase in the performance gap with a higher number of antennas. Moreover, the achieved rates of the low-complexity DNN for rank-k approximation and the DNN for rank-1 approximation get closer while the number of antennas increases due to the accumulated noise in each DNN of the low-complexity architecture.

4) Comparison of Proposed DL-Based Hybrid BF Approach With the State-of-the-Art: We compare the achieved rates of the hybrid BF based on DNN for rank-k matrix approximation presented in Section III-A with conventional hybrid BF algorithms given in [9], [10], [39], an ML-aided hybrid BF algorithm based on CE optimization [25], two DL-based hybrid BF algorithms given in [27], [30], and an autoencoder based hybrid BF algorithm [28] in Figures 17-a, 17-b, and 17-c. In these results, the DNN has 4 convolutional layers and a mini-batch size of 32. We use 2-bit phase shifters in the RF beamformers of 8-by-8 mmW systems, and 3-bit phase shifters in the RF beamformers of 16-by-16 and 32-by-32 mmW systems. The hybrid BF algorithm proposed in [9] designs the beamformers by approximating the channel's dominant singular vectors based on a multi-resolution BF codebook. The proposed multi-resolution codebook has multiple levels, each with BF vectors, which are defined in terms of the set of quantized angles. In [10], a hybrid BF algorithm based on orthogonal matching pursuit (OMP) has been proposed. Authors of [10] formulate the problem as a sparse signal approximation in which near-optimal beamformers are found. The hybrid BF algorithm given in [39] first reduces the set of possible RF-BF vectors based on the dominant beams, which are determined by a metric such as effective power or AoA direction. Then, an exhaustive search is performed over all beamformers in the reduced set. The authors of [25] have proposed an ML-aided hybrid BF algorithm based on adaptive CE optimization. In this algorithm, the weight of each candidate hybrid beamformer is adaptively updated according to their achievable sum-rates and calculates the probability distributions of elements in hybrid beamformers by minimizing the CE. In [27], a DNN is used to predict the BF vectors from received signals using omni beam patterns. In [30], a CNN-based BF technique is proposed to solve the optimum beam-selection problem in Vehicle to Infrastructure (V2I) scenario. In this work, the authors focus on finding the best pair of beams for analog BF of a transceiver with one RF chain and fixed codebooks. Finally, a DNN framework is employed to construct an autoencoder to learn the mapping between the optimum precoder and the multiplication of the analog and digital precoders in [28].

We consider the geometric channel model defined in Section II-D. Figures 17-a, 17-b, and 17-c show the achieved rates of the hybrid BF based on DNN for rank-k matrix approximation, conventional hybrid BF algorithms, a ML-aided hybrid BF algorithm based on CE optimization, two DL-based hybrid BF algorithms, and an autoencoder based hybrid BF algorithm versus SNR for 8-by-8, 16-by-16, and 32-by-32 mmW systems, respectively. It is shown in Figure 17-a that we obtain 23% improvement in rates with the proposed DNN based hybrid BF compared to the ML-based hybrid BF algorithm given in [25] for 8-by-8 mmW



(a) DNN for rank-k approximation. (b) Low-complexity DNN. (c) DNN for rank-1 approximation.

Fig. 18. Achieved rates of the constrained BF and the hybrid BF based on three proposed DNNs with four quantization approaches for the 8-by-8 mmW system.

system when SNR is 20 dB. Since our proposed approach is based on CNNs, which generally perform better than the other ML-based approaches when the input is multi-dimensional, the higher data rate is achieved with our algorithm compared to other data-driven approaches. Moreover, our approach outperforms the other CNN based hybrid BF given in [30], and 26.33% gain in rates is achieved. In the algorithm given in [30], MSE is used as the loss function during the training of CNNs. On the other hand, our proposed loss function minimizes the error between the estimated and real rank-k approximation of the given matrix, which achieves better results compared to MSE. We also achieve 55.3% gain in rates with our approach compared with the conventional hybrid BF algorithm given in [9]. Conventional hybrid BF approaches find the sub-optimal beamformers, while the ML-based hybrid BF algorithms can find the global optimum. Therefore, the performance gain of the proposed approach compared to conventional algorithms further increases.

We see in Figure 17-b that the DNN based hybrid BF achieves 8% gain in rates compared to the CNN based hybrid BF algorithm [30]. Since the performance of the proposed DNN architecture for rank-k approximation degrades with the higher number of antennas, the improvement compared to the CNN based algorithm also decreases. Moreover, the performance of autoencoder based hybrid BF gets better than the DL-based hybrid BF algorithm [27] with a higher number of antennas. However, our approach achieves 21% gain compared to the autoencoder based hybrid BF algorithm. Furthermore, we obtain 34.35% gain in rates compared to the DL-based hybrid BF algorithm [27] for 16-by-16 mmW system. Finally, we achieve 69.5% improvement with our approach compared to the conventional algorithm proposed in [9]. Moreover, the performance of the ML-based hybrid BF algorithm degrades significantly for 16-by-16 mmW system. We observe in Figure 17-c that for 32-by-32 mmW system, our approach achieves 7.9% and 32.68% gains in rates compared to the autoencoder based [28] and the DL-based hybrid BF algorithm [27], respectively. We can see in the results that the gain obtained with the DNN based hybrid BF compared to the DL-based hybrid BF, which uses feedforward neural networks, increases when the number of transmit and receive antennas increases from 8 to 16. The increase in gain occurs since the performance of feedforward neural networks used in the algorithm given in [27] degrades with the larger number of antennas. On the other hand, the performance gap between the autoencoder based hybrid BF and our approach degrades with the higher number of antennas due to the performance loss of the proposed DNN for rank-k approximation with larger sized matrices. Since the proposed loss function is more effective in learning the features of beamformers, the proposed approach achieves 40% gain in rates compared to the CNN based hybrid BF algorithm given in [30] for 32-by-32 mmW system. In these results, we also observe that the gain in achieved rates with the proposed approach increases with SNR for different sized mmW systems. For instance, the improvement in achieved rates with the proposed approach compared to the DL-based algorithm given in [27] is 25% for 16-by-16 mmW system when SNR is -10 dB. On the other hand, the gain in achieved rates compared to the same approach increases to 34.35% when SNR is 20 dB for the same 16-by-16 mmW system. Furthermore, the proposed approach achieves 4.6% gain in rates compared to the autoencoder based BF algorithm given in [28] for 32-by-32 mmW system when SNR is 0 dB. When SNR is 20 dB, the improvement in rates becomes 7.9% for 32-by-32 mmW system.

5) Impact of Different Approaches of Quantization: In this section, we compare the achieved rates of the proposed hybrid BF approaches based on the DNN for rank-k matrix approximation, low-complexity DNN for rank-k matrix approximation, and DNN for rank-1 matrix approximation with four quantization methods proposed in Section IV-A. By using the geometric channel model, we conduct simulations for the 8-by-8 mmW system in which the RF beamformers have 2-bit phase shifters. Figure 18-a shows the achieved rates with the hybrid BF based on DNN for rank-k matrix approximation. We observe that the first and second quantization approaches achieve similar rates and outperform other quantization methods when DNN for rank-k matrix approximation is used. It is shown in 18-b that the third quantization approach achieves the highest data rates with the low-complexity DNN for rank-k matrix approximation. We observe in 18-c that the fourth quantization approach outperforms other methods when the DNN for rank-1 matrix approximation is used. The first approach uses a combination of step and piece-wise linear functions

TABLE III
TIME COMPLEXITY OF DNN BASED HYBRID BF AND STATE-OF-THE-ART

Hybrid BF Technique	Time Complexity
Proposed Hybrid BF Based on DNN for Rank-k Approximation	$O(\max(k^2N_RN_T, k^2N_R^2, k^2N_T^2))$
Proposed Hybrid BF Based on the Low-Complexity DNN for Rank-k Approximation	$O(\max(kN_RN_T, kN_R^2, kN_T^2))$
Proposed Hybrid BF Based on DNN for Rank-1 Approximation	$O(\max(N_RN_T, N_R^2, N_T^2))$
Hybrid BF Based on Adaptive CE [9]	$O\left(\frac{K^2S^3}{L_R}\left(S-\log_K^S\right)\left(N_RN_T^2\right)\right)$
Sparse Hybrid BF [10]	$O\left(N_RN_T^2\right)$
Hybrid BF Based on Efficient Power [39]	$O\left(Q^{L_R}Q^{L_T}\right)$
ML-Aided Hybrid BF Based on CE Optimization [25]	$O\left(I\left(2^{N_q}N_TN_R^2\right)\right)$
DL-Based Hybrid BF via MLP [27]	$O\left(NK_p+N_RN_T\right)$
DL-Based Hybrid BF via CNN [30]	$O(\max(L^2N_RN_T, L^2N_R^2, L^2N_T^2))$
Autoencoder Based Hybrid BF [28]	$O\left(\max(LN_T^2, LN_R^2\right)$

during backward propagation. Therefore, it is differentiable on the regions that are determined by piece-wise linear functions. The second and third approaches use sigmoid function, while the fourth approach sets the derivative of the gradients to the identity matrix during backward propagation. Since these three approaches are differentiable in all regions, they can be learned without suffering the gradient mismatch. We observe that the low-complexity DNN and DNN for rank-1 approximation get the benefit of quantization approaches 2, 3, and 4 more than the DNN for rank-k approximation. The performances of low-complexity DNN and DNN for rank-1 approximation are worse than the DNN for rank-k approximation since they are trained to learn from noisy input data. The results in this section show that quantization approaches 2, 3, and 4 can compensate for this performance degradation more than the quantization approach 1 since the first approach is not differentiable in all regions.

E. Time Complexity Analysis of DNN Based Hybrid BF and State-of-the-Art

It has been shown in Section III-E that the DNN for rank-k approximation has a time complexity of $O(\max(k^2N_RN_T, k^2N_R^2, k^2N_T^2))$ for the constant values of m, n, f, and p, which denote the number of filters, the number of convolutional layers, the spatial size of the filter, and the size of zero-padding, respectively. Moreover, we show in Section III-E that the time complexity of the low-complexity DNN and the DNN for rank-1 approximation is $O(\max(kN_RN_T, kN_R^2, kN_T^2))$ and $O(\max(N_RN_T, N_R^2, N_T^2))$, respectively. Since the DNN consumes the major computation time, the time complexity of the proposed hybrid BF can be approximated with the time complexity of the used DNN architecture. To estimate the mmW channel, the authors of [9] have proposed an algorithm which has a time complexity of $O\left(\frac{K^2S^3}{L_R}\left(S-\log_K^S\right)\right)$, where K, L_R, and S denote the number of BF vectors at the Tx in each stage, the number of RF chains at the Rx, and the number of paths in the channel, respectively. In the hybrid BF algorithm of [9], the SVD of estimated N_R-by-N_T channel matrix is calculated, which has a time complexity of $O(N_RN_T^2)$. With some mathematical manipulations, the time complexity of this algorithm is approximated as $O\left(\frac{K^2S^3}{L_R}\left(S-\log_K^S\right)\left(N_RN_T^2\right)\right)$. Authors of [10] propose an OMP based hybrid BF algorithm with an approximate time complexity of $O\left(N_RN_T^2\right)$. In [39], Q Tx and Q Rx beams with the largest effective powers are selected, and an exhaustive search is performed on the reduced set. The time complexity of this algorithm is approximated as $O\left(Q^{L_R}Q^{L_T}\right)$. In the ML-aided hybrid BF algorithm proposed in [25], effective channel matrices and corresponding BB beamformers are computed for C_N candidate beamformers. This part of the algorithm, which has the highest time complexity, is $O\left(C_NN_TN_R^2\right)$. The number of candidate beamformers C_N should be selected at least 2^{N_q}, where N_q is the number of bits used in the phase shifters, to cover the all possible beam directions. After I iterations, the total time complexity of the algorithm given in [25] becomes $O\left(I\left(2^{N_q}N_TN_R^2\right)\right)$. In the training phase of the DL-based hybrid BF algorithm introduced in [27], the Rx receives training symbols using N_{tr} different RF-BF vectors, which leads to time complexity of $O\left(N_RN_TN_{tr}\right)$. Finally, a fully-connected feed-forward DNN is trained with N pilot sequences each of with a length of K_p. Since the complexity of the DNN is approximately linear in the size of the input, which is NK_p for the DNN in this algorithm, the time complexity of training and prediction phases are found to be $O\left(NK_p+N_RN_TN_{tr}\right)$ and $O\left(NK_p+N_RN_T\right)$, respectively. In [28], the matrix multiplication of the analog and digital beamformers dominates the computation time. We assume that $\mathbf{T}_A \in \mathbb{C}^{L_T\times N_T}$ and $\mathbf{T}_D \in \mathbb{C}^{L\times L_T}$ are the analog and digital precoders, respectively. Moreover, $\mathbf{R}_A \in \mathbb{C}^{N_R\times L_R}$ and $\mathbf{R}_D \in \mathbb{C}^{L_R\times L}$ are considered to be the analog and digital combiners, respectively. Then, the time complexity of this algorithm becomes $O\left(\max(LN_T^2, LN_R^2\right)$. In [30], a CNN with four convolutional layers and one fully-connected layer is used. Moreover, the number of filters in each convolutional layer, the spatial size of each filter, and the size of zero-padding are kept the same with the DNN for rank-k approximation. The number of output nodes in the fully connected layer is $k(2N_R+2N_T+1)$ as in the DNN for rank-k approximation. Therefore, the time complexity of the algorithm given in [30] can be approximated as $O(\max(L^2N_RN_T, L^2N_R^2, L^2N_T^2))$. In the given time complexities, L, N_T, and N_R denote the number of data streams,

TABLE IV
RUN TIME RESULTS FOR DNN BASED HYBRID BF AND STATE-OF-THE-ART

Size	DNN for Rank-k Approximation	Adaptive CE [9]	Sparse BF [10]	Efficient Power [39]	ML-Aided BF [25]	BF via MLP [27]	BF via CNN [30]	BF via Autoencoder [28]
4-by-4	0.313 sec	0.066 sec	0.0024 sec	0.162 sec	0.0136 sec	0.0645 sec	0.1597 sec	0.0405 sec
8-by-8	0.365 sec	0.468 sec	0.0038 sec	0.335 sec	0.0175 sec	0.0705 sec	0.447 sec	0.054 sec
16-by-16	0.526 sec	3.885 sec	0.006 sec	0.453 sec	0.0321 sec	0.0853 sec	0.932 sec	0.0764 sec

the number of transmit antennas, and the number of receive antennas, respectively. The complexities of testing phases for the proposed approach and other state-of-the-art approaches are given in Table III.

Based on the given approximations for the time complexities of our approach and the state-of-art algorithms, the DL-based hybrid BF algorithm given in [27] achieves the least time complexity for small values of N and K_p. For large values of N and K_p, the time complexity of the DNN for rank-1 approximation becomes smaller than [27]. Furthermore, the DNN for rank-k approximation and the low-complexity DNN have comparable time complexities to the DNN for rank-1 approximation for the small values of k. On the other hand, the time complexity of the hybrid BF algorithms given in [10] and [28] are less than the hybrid BF algorithms given in [9] and [39]. Moreover, [28] has a smaller time complexity than the sparse hybrid BF algorithm when the number of data streams (L) is less than $\min(N_R, N_T)$. The time complexity of the ML-aided hybrid BF algorithm proposed in [25] grows exponentially with the number of bits used in the phase shifters. Therefore, the time complexity of this algorithm is comparable to the complexity of the sparse hybrid BF algorithm when the number of bits is small. However, it starts to increase rapidly while the number of bits grows. Finally, we measure the average run time of our approach, and the state-of-art hybrid BF algorithms over a batch size equal to 256, as shown in Table IV. In our simulations, the values of Q, N_q, K, and S are set to 4, 3, 2, and 1, respectively. For the scope of these results, the values of k, L, N_R, N_T, L_R, L_T, and NK_p are picked as either 4, 8, or 16 such that $k = L = N_R = N_T = L_R = L_T = NK_p$.

V. CONCLUSION

In this paper, we first presented three DNN architectures with different levels of complexity to learn the SVD operation. The first architecture estimates the k most significant singular values and singular vectors of the given matrix. The second architecture uses k low-complexity DNNs in which each DNN predicts one singular value and one singular vector. The third architecture estimates the k largest singular values and singular vectors of the given matrix by using a single DNN recursively. Finally, we introduced a DNN based hybrid BF approach to design the analog and digital beamformers under the constraints of the finite-precision phase shifters and the power constraint. Extensive simulations were presented to evaluate the performance of the proposed DNN architectures for the SVD. By using the geometric channel model, we compared the achieved rates of the DNN based hybrid BF approach with the unconstrained BF, three conventional hybrid BF algorithms, an ML-aided hybrid BF algorithm, two DL-based hybrid BF algorithms, and an autoencoder based hybrid BF algorithm. With our simulation results, we show that the DNN based hybrid BF approach obtains up to $50 - 70\%$ gain in rates compared to conventional hybrid BF algorithms, and it achieves $10 - 30\%$ improvement compared to ML-based hybrid BF algorithms. In the future, we aim to explore the unsupervised techniques such as generative adversarial networks (GANs) for the SVD and hybrid BF to eliminate the need for supplying valid singular values and singular vectors, which leads to reduced overhead.

REFERENCES

[1] F. Boccardi, R. W. Heath, Jr., A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[2] L. Kong, M. K. Khan, F. Wu, G. Chen, and P. Zeng, "Millimeter-wave wireless communications for IoT-cloud supported autonomous vehicles: Overview, design, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 62–68, Jan. 2017.

[3] M. R. Palattella *et al.*, "Internet of Things in the 5G era: Enablers, architecture, and business models," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 2016.

[4] S. Han, C.-L. I, Z. Xu, and C. Rowell, "Large-scale antenna systems with hybrid analog and digital beamforming for millimeter wave 5G," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 186–194, Jan. 2015.

[5] T. S. Rappaport *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.

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

[7] A. F. Molisch *et al.*, "Hybrid beamforming for massive MIMO: A survey," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 134–141, Sep. 2017.

[8] Y. Ren, Y. Wang, C. Qi, and Y. Liu, "Multiple-beam selection with limited feedback for hybrid beamforming in massive MIMO systems," *IEEE Access*, vol. 5, pp. 13327–13335, 2017.

[9] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, Jr., "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 831–846, Oct. 2014.

[10] O. E. Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi, and R. W. Heath, Jr., "Spatially sparse precoding in millimeter wave MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1499–1513, Mar. 2014.

[11] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, Jr., "Hybrid precoding for millimeter wave cellular systems with partial channel knowledge," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2013, pp. 1–5.

[12] A. Alkhateeb, G. Leus, and R. W. Heath,Jr., "Limited feedback hybrid precoding for multi-user millimeter wave systems," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 6481–6494, Nov. 2015.

[13] T. E. Bogale and L. B. Le, "Beamforming for multiuser massive MIMO systems: Digital versus hybrid analog-digital," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 4066–4071.

[14] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[15] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based MIMO communications," *CoRR*, vol. abs/1707.07980, pp. 1–9, Jul. 2017.

[16] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[17] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8549–8560, Sep. 2018.

[18] H. Huang *et al.*, "Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 214–222, Feb. 2020.

[19] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Math.*, vol. 14, no. 5, pp. 403–420, Apr. 1970.

[20] E. Angerson *et al.*, "LAPACK: A portable linear algebra library for high-performance computers," in *Proc. SUPERCOMPUTING*, 1990, pp. 2–11.

[21] A. Cichocki, "Neural network for singular value decomposition," *Electron. Lett.*, vol. 28, no. 8, pp. 784–786, Apr. 1992.

[22] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, Jan. 1989.

[23] H. Huang, Y. Peng, J. Yang, W. Xia, and G. Gui, "Fast beamforming design via deep learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1065–1069, Jan. 2020.

[24] T. Lin and Y. Zhu, "Beamforming design for large-scale antenna arrays using deep learning," *IEEE Wireless Commun. Lett.*, vol. 9, no. 1, pp. 103–107, Jan. 2020.

[25] X. Gao, L. Dai, Y. Sun, S. Han, and I. Chih-Lin, "Machine learning inspired energy-efficient hybrid precoding for mmWave massive MIMO systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[26] Y. Long, Z. Chen, J. Fang, and C. Tellambura, "Data-Driven-Based analog beam selection for hybrid beamforming under mm-wave channels," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 2, pp. 340–352, May 2018.

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

[28] H. Huang, Y. Song, J. Yang, G. Gui, and F. Adachi, "Deep-learning-based millimeter-wave massive MIMO for hybrid precoding," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3027–3032, Mar. 2019.

[29] K. Satyanarayana, M. El-Hajjar, A. A. M. Mourad, and L. Hanzo, "Multi-user hybrid beamforming relying on learning-aided link-adaptation for mmWave systems," *IEEE Access*, vol. 7, pp. 23197–23209, 2019.

[30] A. Klautau, P. Batista, N. Gonzalez-Prelcic, Y. Wang, and R. W. Heath, Jr., "5G MIMO data for machine learning: Application to beam-selection using deep learning," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2018, pp. 1–9.

[31] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[32] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[33] Y. L. Cun *et al.*, "Handwritten digit recognition with a back-propagation network," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1989, pp. 396–404.

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

[35] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4674–4683.

[36] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, pp. 1–14, Mar. 2018.

[37] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2018, pp. 1–21.

[38] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, pp. 1–12, Aug. 2013.

[39] J. Singh and S. Ramakrishna, "On the feasibility of codebook-based beamforming in millimeter wave systems with multiple antenna arrays," *IEEE Trans. Wireless Commun.*, vol. 14, no. 5, pp. 2670–2683, May 2015.

[40] S. Zhou and G. B. Giannakis, "Optimal transmitter eigen-beamforming and space-time block coding based on channel mean feedback," *IEEE Trans. Signal Process.*, vol. 50, no. 10, pp. 2599–2613, Oct. 2002.

[41] M. Stojnic, H. Vikalo, and B. Hassibi, "Rate maximization in multi-antenna broadcast channels with linear preprocessing," *IEEE Trans. Wireless Commun.*, vol. 5, no. 9, pp. 2338–2342, Sep. 2006.

[42] S. A. Jafar, S. Vishwanath, and A. Goldsmith, "Channel capacity and beamforming for multiple transmit and receive antennas with covariance feedback," in *Proc. IEEE Int. Conf. Communications. Conf. Rec. (ICC)*, vol. 7, Jun. 2001, pp. 2266–2270.

[43] M. R. Akdeniz *et al.*, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1164–1179, Jun. 2014.

[44] T. S. Rappaport, Y. Qiao, J. I. Tamir, J. N. Murdock, and E. Ben-Dor, "Cellular broadband millimeter wave propagation and angle of arrival for adaptive beam steering systems (invited paper)," in *Proc. IEEE Radio Wireless Symp.*, Jan. 2012, pp. 151–154.

[45] A. A. M. Saleh and R. Valenzuela, "A statistical model for indoor multipath propagation," *IEEE J. Sel. Areas Commun.*, vol. SAC-5, no. 2, pp. 128–137, Feb. 1987.

[46] A. F. Molisch, "A generic model for MIMO wireless propagation channels in Macro-and microcells," *IEEE Trans. Signal Process.*, vol. 52, no. 1, pp. 61–71, Jan. 2004.

[47] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[48] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[49] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. Int. Conf. Learn. Representations (ICLR)*, May 2016, pp. 1–14.

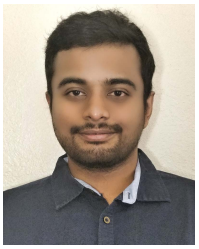
[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, May 2015, pp. 1–15.

[51] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. (OSDI)*, Nov. 2016, pp. 265–283.

[52] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5353–5360.

[53] F. Branchaud-Charron, F. Rahman, and T. Lee. *Keras*. [Online]. Available: https://github.com/keras-team

Ture Peken received the B.S. degree in telecommunications and computer engineering from Istanbul Technical University, Turkey, in 2011, and the M.S. degree in electrical engineering: systems from the University of Michigan, Ann Arbor, in 2012. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ, USA. She joined The University of Arizona as a Graduate Assistant in 2014. Since 2019, she has been with Keysight Technologies, Santa Rosa, CA, USA. Her current research interests include wireless communications, millimeter-wave systems, machine learning and its applications to beamforming, and massive MIMO systems.



Sudarshan Adiga (Student Member, IEEE) received the B.E. degree in telecommunication engineering from the Ramaiah Institute of Technology, Bengaluru, in 2015, and the M.S. degree in electrical and computer engineering from The University of Arizona, Tucson, AZ, USA, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His current research interests include machine learning, information theory, and wireless communications.

Ravi Tandon (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology at Kanpur (IIT Kanpur) in 2004 and the Ph.D. degree in electrical and computer engineering from the University of Maryland, College Park (UMCP), in 2010. Prior to joining The University of Arizona in Fall 2015, he was a Research Assistant Professor at Virginia Tech with positions in the Bradley Department of Electrical and Computer Engineering (ECE), Hume Center for National Security and Technology, and the Discovery Analytics Center, Department of Computer Science. From 2010 to 2012, he was a Post-Doctoral Research Associate with Princeton University. He is currently an Assistant Professor with the Department of ECE, The University of Arizona. He serves as an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. His current research interests include information theory and its applications to wireless networks, communications, security and privacy, machine learning, and data mining. He was a recipient of the 2018 Keysight Early Career Professor Award, NSF CAREER Award in 2017, and the Best Paper Award at IEEE GLOBECOM 2011.

Tamal Bose received the Ph.D. degree from Southern Illinois University.

He was the Founder and the National Director of the NSF sponsored multi-university multi-industry wireless research center called the Broadband Wireless and Applications Center from 2012 to 2017. From 2007 to 2013, he was a Tenured Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Tech. He served as an Associate Director of Wireless@VT. In 2008, he established the NSF sponsored WICAT@VT site and served as the Site Director. He also served as the Department Head of electrical and computer engineering with Utah State University, Logan, Utah. There, he founded and directed the Center for High-Speed Information Processing. He is also the Department Head and a Professor with the Electrical and Computer Engineering Department, The University of Arizona, Tucson. His current research projects are funded by the National Science Foundation (NSF), the Department of Energy (DOE), the Defense Intelligence Agency (DIA), and some private companies. Throughout his career, he has directed research and worked as the Principal Investigator (PI) and Co-PI of research funded by NSF, DOE, DIA, Defense Advanced Research Program Administration (DARPA), Office of Naval Research (ONR), Missile Defense Agency (MDA), Air Force Research Laboratory (AFRL), NASA, and over 15 different private companies. The total volume of research, he has conducted as PI/Co-PI exceed 20M. He is the author/coauthor of over 70 journal articles and over 100 conference papers. He is the author of the text *Digital Signal and Image Processing*, (John Wiley, 2004) and a coauthor of *Basic Simulation Models of Phase Tracking Devices Using MATLAB*, (Morgan and Claypool Publishers, 2010). His research interests include signal detection and classification for cognitive radios, channel equalization, adaptive filtering algorithms, and nonlinear effects in digital filters.