Distributed Deep Convolutional Compression for Massive MIMO CSI Feedback

Qianqian Yang, Mahdi Boloursaz Mashhadi and Deniz Gündüz
Dept. of Electrical and Electronic Eng., Imperial College London, UK
Email: {q.yang14, m.boloursaz-mashhadi, d.gunduz}@imperial.ac.uk

Abstract

Massive multiple-input multiple-output (MIMO) systems require downlink channel state information (CSI) at the base station (BS) to achieve spatial diversity and multiplexing gains. In a frequency division duplex (FDD) multiuser massive MIMO network, each user needs to compress and feedback its downlink CSI to the BS. The CSI overhead scales with the numbers of antennas, users and sub-carriers, and becomes a major bottleneck for the overall spectral efficiency. In this paper, we propose a deep learning (DL)-based CSI compression scheme, called *DeepCMC*, composed of convolutional layers followed by quantization and entropy coding blocks. In comparison with previous deep learning DL-based CSI reduction structures, DeepCMC includes quantization and entropy coding blocks and minimizes a weighted rate-distortion cost which enables a trade-off between the CSI quality and its feedback overhead. Simulation results demonstrate that DeepCMC outperforms the state of the art CSI compression schemes in terms of the reconstruction quality of CSI for the same compression rate. We also propose a distributed version of DeepCMC for a multi-user MIMO scenario to encode and reconstruct the CSI from multiple users in a distributed manner. Distributed DeepCMC not only utilizes the inherent CSI structures of a single MIMO user for compression, but also benefits from the correlations among the channel matrices of nearby users to further improve the performance in comparison with DeepCMC.

I. Introduction

Massive multiple-input multiple-output (MIMO) systems are considered as the main enabler of 5G and future wireless networks thanks to their ability to serve a large number of users

This work was supported by the European Research Council (ERC) through project BEACON (grant no 677854). Part of this work was presented at the IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), Pittsburg, PA, Oct. 2019 [1].

simultaneously, achieving impressive levels of energy and spectral efficiency. The base station (BS) in a massive MIMO setting relies on the downlink channel state information (CSI) to fully benefit from the available degrees of freedom and achieve the promised performance gains [2], [3]. In time division duplex (TDD) mode of operation, massive MIMO systems can exploit the uplink CSI for downlink transmission, thanks to channel reciprocity. On the other hand, frequency division duplex (FDD) operation is more desirable due to better coverage it provides; however, channel reciprocity does not hold in FDD; and hence, downlink CSI must be estimated at user equipments (UEs) during the training period and fed back to the BS.

The resulting feedback overhead becomes excessive due to the massive number of antennas and users being served, and has motivated various CSI reduction techniques based on vector quantization [4] and compressed sensing (CS) [5], [6]. In vector quantized CSI feedback, the overhead scales linearly with system dimensions, which becomes restrictive in many practical massive MIMO scenarios. On the other hand, CS-based approaches rely on sparsity of the CSI data in a certain transform domain, which may not represent the channel structure accurately for many practical MIMO scenarios. CS-based approaches are also iterative, which introduces additional delay.

Following the recent resurgence of machine learning, and more specifically deep learning (DL) techniques for physical layer communications [7], [8], DL-based MIMO CSI estimation, compression and feedback techniques have recently been proposed [9], [10]. The DL-based CSI compression scheme, CSINet [11], showed significant improvement over previous works that utilized compressive sensing and sparsifying transforms. Following CSINet, several subsequent schemes were proposed which use autoencoder architectures to reduce the MIMO CSI feedback overhead by learning low-dimensional features of the channel gain matrix from training data [1], [11]–[21]. In [13], the authors improve CSINet by utilizing a recurrent neural network to utilize temporal correlations in time-varying channels. Utilizing bi-directional channel reciprocity, the authors in [14] use the uplink CSI as an additional input to further improve the results utilizing the correlation between downlink and uplink channels.

Several autoencoder-based CSI reduction techniques [11], [12], [14] focus on dimensionality reduction by direct application of the autoencoder architecture. These works are based on the assumption that reducing the dimension of the CSI matrix to be fed back to the BS would result in reduced feedback overhead. However, in general, the reduced dimension CSI matrix does not result in the most efficient representation, and it can be further compressed by efficient

quantization and compression techniques. Design of efficient compression techniques and the impact of such compression on the CSI reconstruction accuracy has not been considered in [11], [12], [14]. The authors in [20] use uniform quantization on the reduced CSI values. However, the distribution of the output of the encoder neural network is not uniform and uniform quantization shall produce values that are not equally probable, and can be further compressed. Considering this, the authors in [18] used non-uniform μ-law quantization to get more evenly distributed quantized symbols. A DL-based architecture is proposed in [21], [22] to learn a non-uniform quantizer.

In this paper, we propose a DL-based CSI compression scheme, called DeepCMC, composed of a fully convolutional autoencoder structure in conjunction with quantization and entropy coding blocks. This is the first work [1] for MIMO CSI compression that uses an estimate of the local probability distribution of the quantized autoencoder output to efficiently compress it by a context-adaptive arithmetic entropy coder at rates closely approaching its entropy. Following our initial work, arithmetic entropy coding is also adopted by [21] for CSI compression. Here, we also propose a novel distributed DeepCMC architecture to encode the CSI from multiple users in a distributed manner, which are decoded jointly at the BS. Our goal is to exploit the correlations among the CSI matrices of nearby users to further reduce the required communication overhead. Note that a major benefit of a massive MIMO BS is its ability to simultaneously serve a large number of users in its coverage area. Hence, exploiting common structures and correlations among the channel matrices of the users to better compress their CSI can significantly improve the overall spectral efficiency.

In comparison with the previous DL-based CSI compression techniques, the main contributions of the proposed DeepCMC and its distributed version can be summarized as follows:

i) Existing DL-based architectures for CSI compression all include a fully connected layer, which means that they can only be utilized for a specified input size, e.g., for a given number of OFDM sub-carriers. This would mean that a different NN needs to be trained for every different resource allocation setting, and users need to store NN coefficients for all these networks, limiting the practical implementation of these solutions. On the other hand, the proposed DeepCMC architecture is fully convolutional, and has no densely connected layers, which makes it flexible for a wider range of MIMO scenarios. Our simulations show that the convolutional kernels of DeepCMC, once trained, work sufficiently well for a large range of sub-carriers.

ii) Many of the existing DL-based architectures for CSI compression mainly focus on di-

mensionality reduction by direct application of the autoencoder architecture and do not consider further compression of the CSI at a bit level [11], [12], [14]. DeepCMC includes quantization and entropy coding blocks within its architecture to directly convert the channel gain matrix into bits for subsequent communication. In contrast to previous works that minimize the reconstruction mean square error (MSE) of the reconstructed CSI matrix, DeepCMC minimizes a weighted rate-distortion cost that takes into account both the compression rate (in terms of bits per CSI value) and the reconstruction MSE, which significantly improves the performance and enables a rate-distortion trade-off. Although uniform and non-unifrom μ-law quantization are considered in [20] and [18], respectively, the quantization process is still blind to the specific distribution of the reduced CSI values. However, our proposed DeepCMC scheme learns the local probability distributions of the quantizer output and uses it in conjunction with context-adaptive arithmetic entropy coding to efficiently compress the quantizer output at rates closely approaching its entropy.

iii) We propose distributed DeepCMC for a multi-user massive MIMO scenario such that different users compress their CSI in a distributed manner while the BS jointly reconstructs the CSI of multiple users from the received feedback messages. This is motivated by the information theoretic results on distributed lossy compression of correlated sources [23], and is based on the fact that the CSI of nearby users are correlated as they share common multi-path components from scatterers located far away from them. Hence, distributed DeepCMC not only utilizes the inherent CSI structures of a single MIMO user for compression, but also benefits from the channel correlations among nearby MIMO users to further improve the performance. To the best of our knowledge, distributed DeepCMC is the first distributed NN architecture for multi-user MIMO CSI compression.

This paper is organized as follows. In Section II, we present the system model. In Sections III and IV we present our proposed DeepCMC scheme for massive MIMO CSI compression and its distributed version, respectively. Section V provides the simulation results and Section VI concludes the paper.

II. System Model

We consider a massive MIMO setting in which a BS with N_t antennas serves K single-antenna users utilizing orthogonal frequency division multiplexing (OFDM) over N_c subcarriers. We denote by $\mathbf{H}^k \in \mathbb{C}^{N_c \times N_t}$ the downlink channel matrix for user k, and by $\mathbf{v}^k \in \mathbb{C}^{N_t \times 1}$ the

precoding vector used for downlink transmission to user k. The received signal at user k is given by

$$\mathbf{y}^k = \mathbf{H}^k \mathbf{v}^k x^k + \mathbf{H}^k \sum_{i \neq k} \mathbf{v}^i x^i + \mathbf{z}^k, \tag{1}$$

where $x^k \in \mathbb{C}$ are the data-bearing symbols, and $\mathbf{z}^k \in \mathbb{C}^{N_c \times 1}$ is the additive noise vector, for $k \in [K] \triangleq \{1, ..., K\}$. In order to design the precoding vectors $\mathbf{v}^k$ for efficient transmission, the BS requires estimates of the downlink CSI matrices, $\mathbf{H}^k$. To this end, in an FDD system, each user estimates its downlink CSI matrix through pilot-based training, and transmits the estimated CSI back to the BS. Hence, the overhead for CSI feedback from the users grows with $K \times N_c \times N_t$, and becomes prohibitive for wideband massive MIMO systems where K, N_c and N_t are large.

To cope with this challenge, the users need to efficiently compress their channel matrices $\mathbf{H}^k$. Let $\mathbf{H}^k = [\mathbf{h}_1^k, \mathbf{h}_2^k, \ldots, \mathbf{h}_{N_c}^k]^T$, where $\mathbf{h}_n^k \in \mathbb{C}^{N_t}$ is the channel gain vector of user k over subcarrier n, $n \in [N_c]$. Assume that the BS is equipped with a uniform linear array (ULA) with response vector $\mathbf{a}(\phi) = [1, e^{-j\frac{2\pi d}{\lambda}\sin\phi}, \cdots, e^{-j\frac{2\pi d}{\lambda}(N_t-1)\sin\phi}]^T$, where ϕ is the angle of departure (AoD), and d and λ denote the distance between adjacent antennas and carrier wavelength, respectively. The channel gain vectors are a summation of multipath as

$$\mathbf{h}_n^k = \sqrt{\frac{N_t}{L^k}} \sum_{l=1}^{L^k} \alpha_l^k e^{-j2\pi\tau_l^k f_s \frac{n}{N_c}} \mathbf{a}(\phi^k), \tag{2}$$

where L^k is the number of downlink multipath components for user k with τ_l^k and α_l^k denoting the corresponding delay and propagation gain for the components, respectively, and f_s is the sampling rate. According to (2), the CSI values for nearby sub-channels, antennas and users are correlated due to similar propagation paths, gains, delays and AoDs. This correlation can be exploited to compress the CSI and reduce the feedback overhead.

Designing practically efficient codes for lossy compression is challenging even for memoryless sources with explicitly defined distribution models. Here, we take an alternative data-driven approach and propose a deep NN architecture, called DeepCMC, which learns the compression scheme when trained over large datasets of channel matrices. DeepCMC uses CNN layers and entropy coding blocks to learn the CSI compression scheme that can best leverage the underlying correlations.

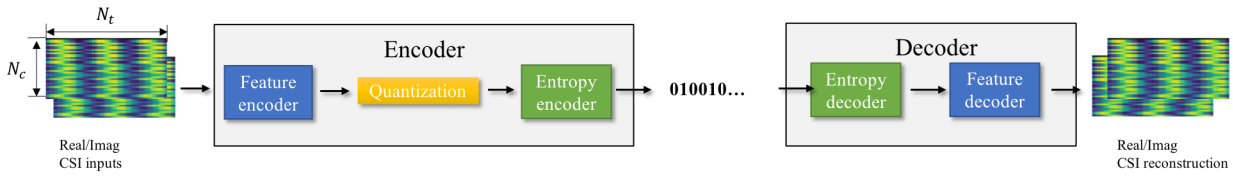

Fig. 1: The encoder/decoder architecture for the proposed CSI feedback compression scheme DeepCMC.

For the general case of K users, we have a multi-terminal lossy source coding problem [23], where our goal is to compress correlated CSI matrices from different users in a distributed manner and at an acceptable distortion and complexity. As opposed to the single user setting, this problem is elusive even in the ideal information theoretic setting. The general solution is known only for jointly Gaussian source distributions under squared error distortion [23], [24], or for discrete memoryless sources under log-loss as the distortion measure [25]. Here, we propose a deep NN architecture, called distributed DeepCMC, and train it over a large dataset of channel matrices to achieve a distributed CSI compression scheme in a data-driven manner without explicit knowledge of the underlying distributions. Distributed DeepCMC leverages the correlations among the CSI of multiple users to further improve the rate-distortion performance in comparison with separate DeepCMC architectures for each user.

III. DeepCMC

In this section, we present our proposed NN architecture, DeepCMC, for encoding and subsequent reconstruction of downlink CSI for a single massive MIMO user. This will be extended to the multiple-user MIMO scenario in Section IV. The overview of our proposed model architecture for DeepCMC is shown in Fig. 1, where the two channel inputs represent the real and imaginary parts of the channel matrix. The user compresses its CSI into a variable length bit stream. The encoder comprises a CNN-based feature encoder, a uniform element-wise scalar quantizer, and an entropy encoder. The feature encoder extracts key features from the CSI matrix to obtain a lower dimensional representation, which is subsequently converted into a discrete-valued vector by applying scalar quantization. While previous works simply send the 32-bit scalar quantized version of the feature vector as the CSI feedback [11], [12], [14], we have observed that the autoencoder structure does not produce uniformly distributed feature values, and hence, can be further compressed.

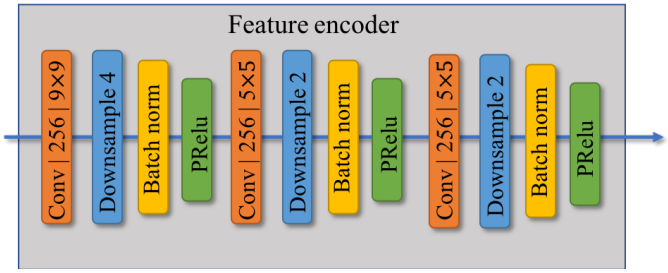

Fig. 2: Feature encoder architecture.

To further reduce the required feedback, we employ an entropy encoder; in particular, we use the context-adaptive binary arithmetic coding (CABAC) technique [26], which outputs a variable-length bit stream. Upon receiving this CSI-bearing bit stream, the BS first processes it by an entropy decoder to reproduce the lower-dimensional representation of the CSI feedback. This representation is then input to the feature decoder NN to reconstruct the estimated channel gain matrix. We present each component of our proposed model in more detail below.

A. Feature encoder and decoder

The CNN architecture used for the feature encoder and decoder are presented in Fig. 2 and Fig. 3, respectively, where Conv$|256|9 \times 9$ represents a convolutional layer with 256 kernels, each of size 9×9. The feature encoder consists of three convolutional layers, the first of which uses kernels of size 9×9, and the other two use kernels of size 5×5. The "SAME" padding technique is used, such that the input and output of each convolutional layer have the same size (the number of channels vary). Each convolutional layer is followed by downsampling to reduce dimensionality. We use PRelu as the activation function, and apply batch normalization to each layer. Let $\mathbf{M} = f_{\mathrm{f-en}}(\mathbf{H}, \mathbf{\Theta}_{en})$, where $f_{\mathrm{f-en}}$ denotes the feature encoder at the user, and $\mathbf{\Theta}_{en}$ denotes its parameter vector. $\mathbf{M}$ consists of 256 feature maps of size $\frac{N_t}{16} \times \frac{N_c}{16}$. Note that this fully convolutional architecture allows us to use the same encoder network for any number of transmit antennas and subcarriers, while the feature vector dimension depends on the input size, which allows us to scale the CSI feedback volume with the channel dimension.

The feature decoder at the BS performs the corresponding inverse operations, consisting of convolutional and upsampling layers. At the BS, the output of the entropy decoder is fed into the feature decoder to reconstruct the channel gain matrix. Similarly to the feature encoder, the

decoder includes three layers of convolutions (with the same kernel sizes as the encoder) and upsampling (inverse of the downsampling operation at the encoder). The decoder architecture also includes two residual blocks with shortcut connections that skip several layers with $+$ denoting element-wise addition in Fig. 3. This structure eases the training of the network by preventing vanishing gradient along the stacked non-linear layers [27]. To enable this, the input and output of a residual block must have the same size. Each residual block comprises two convolutional layers (normalized using the batch norm) and uses PRelu as the activation function. Inspired by [28], we also use an identical shortcut connecting the input and output of the residual blocks, which improves the performance as revealed by the experiments. Let $\widehat{\mathbf{H}} = f_{\mathrm{f-de}}(\widehat{\mathbf{M}}, \mathbf{\Theta}_{de})$ denote the output of the joint decoder, parameterized by $\mathbf{\Theta}_{de}$, and $\widehat{\mathbf{M}}$ denote the estimate of $\mathbf{M}$ provided by the entropy decoder. $\widehat{\mathbf{H}}$ denotes the reconstructed CSI matrix at the BS.

B. Quantization and Entropy coding

A major contribution of our proposed model in comparison with the existing DNN architectures for CSI compression in the literature [11], [12], [14] is the inclusion of the entropy coding block, which encodes quantized CSI data into bits at rates closely approaching its entropy.

Quantization is performed by a uniform scalar quantizer denoted by f_q, which quantizes each element of $\mathbf{M}$ to the closest integer. We denote the quantized output as $\overline{\mathbf{M}} = f_q(\mathbf{M})$.

The entropy encoder converts the quantized values in $\overline{\mathbf{M}}$ into bit streams using CABAC [26] based on the input probability model learned during training. Let $\mathbf{s} = f_{\mathrm{e-en}}(\overline{\mathbf{M}}, P)$ denote the bit stream derived by passing $\overline{\mathbf{M}}$ through the entropy coder, denoted by $f_{\mathrm{e-en}}$, where P is the probability density function, estimated during training, as it will be described later in the following subsection.

The estimate of $\mathbf{M}$, denoted by $\widehat{\mathbf{M}}$, is recovered at the BS by decoding the received codeword $\mathbf{s}$ using the corresponding entropy decoder as $\widehat{\mathbf{M}} = f_{\mathrm{e-de}}(\mathbf{s}, P)$. Finally, $\widehat{\mathbf{M}}$ is fed into the feature decoder to reconstruct the CSI matrix. Note that the scalar uniform quantizer followed by arithmetic entropy coding (CABAC) in our DeepCMC architecture acts as an adaptive variable bit-depth quantizer that optimally encodes the input at rates closely approaching its entropy. This alleviates the need to design more complex non-uniform quantizer blocks that optimize the quantizer thresholds according to the input distribution as proposed in [21], [22].

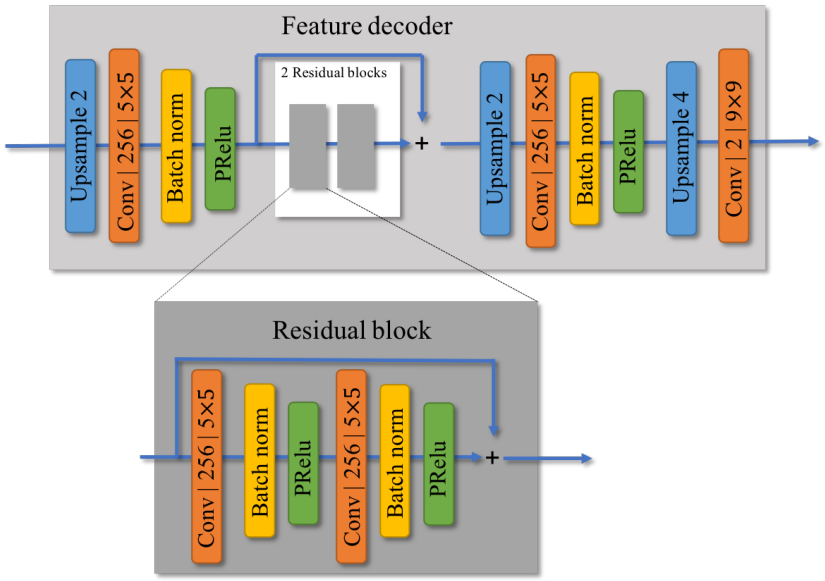

Fig. 3: Feature decoder architecture.

C. *Optimization*

As quantization is not a differentiable function, it cannot be implemented within the gradient-based optimization framework. To overcome this, we replace the uniform scalar quantizer with independently and identically distributed (i.i.d) uniform noise during training. Hence, denoting the quantization noise vector by $\Delta\mathbf{M}$ with i.i.d elements from $U[0,1]$, we approximate the quantized feature matrix by $\widetilde{\mathbf{M}} = \mathbf{M} + \Delta\mathbf{M}$.

Now denote by $P(\widetilde{\mathbf{M}}, \boldsymbol{\Theta}_p)$, the probability density function for $\widetilde{\mathbf{M}}$ specified by the set of parameters $\boldsymbol{\Theta}_p$, which is estimated during training similarly to [29]. Our loss function is given by

$$L(\boldsymbol{\Theta}_{en}, \boldsymbol{\Theta}_{de}, \boldsymbol{\Theta}_p) = \mathbb{E}_{\mathbf{H},\Delta\mathbf{M}}\Bigg(-\frac{1}{N_c N_t}\log P(f_{\text{f-en}}(\mathbf{H}, \boldsymbol{\Theta}_{en}) + \Delta\mathbf{M}, \boldsymbol{\Theta}_p) + \lambda \text{MSE}\Big(f_{\text{f-de}}\big(f_{\text{f-en}}(\mathbf{H}, \boldsymbol{\Theta}_{en}) + \Delta\mathbf{M}, \boldsymbol{\Theta}_{de}\big), \mathbf{H}\Big)\Bigg), \tag{3}$$

where

$$\text{MSE}\Big(\widehat{\mathbf{H}}, \mathbf{H}\Big) = \frac{1}{N_c N_t}\|\mathbf{H} - \hat{\mathbf{H}}\|_2^2,$$

and the expectation is over the training set of channel matrices and the quantization noise. During training, the entropy of the quantizer outputs, estimated by the trainable probability model, is jointly minimized with the reconstruction MSE by optimizing the parameters for both the probability model and the autoencoder. By utilizing the entropy coding block with the optimized probability model, the actual bit rate of the encoder output closely approximates this entropy. More precisely, the first part of the loss function in (3) represents the entropy of the feedback data, or equivalently the size of the feedback in bits that must be transmitted, while the second part is the weighted MSE of the reconstructed channel gain matrices. Hence, training $\mathbf{\Theta}_{en}, \mathbf{\Theta}_{de}$ and $\mathbf{\Theta}_{p}$ values, which parameterize the feature encoder, the feature decoder, and the probability models, respectively, minimizes the feedback overhead and the reconstruction loss, simultaneously.

The λ value governs the trade-off between the compression rate and the reconstruction loss. A larger λ leads to a better reconstruction but a higher feedback overhead and vice versa. In order to recover the trade-off between the compression rate and the reconstruction loss, we train DeepCMC with different λ values. For a small λ value, the network tries to reduce the feedback rate, while as λ increases, it tries to keep the MSE under control while slightly increasing the rate. After training, each λ value specifies a set of parameters $\mathbf{\Theta}_{en}, \mathbf{\Theta}_{de}, \mathbf{\Theta}_{p}$. By selecting the λ value according to user's requirements in terms of CSI quality and the available feedback capacity, we can obtain the encoder and decoder parameters with the best performance under these constraints. This would require the user and the BS to have a list of encoder/decoder parameters to be used for different rate-MSE quality trade-offs, and the user to send the λ value together with the encoded bitstream $\mathbf{s}$ to the BS, so that the BS employs the matching decoder parameters.

We emphasize here that the feature encoder and decoder networks are fully convolutional, and do not include any fully connected layers. Moreover the implemented entropy code can operate on inputs of any size. Therefore, the DeepCMC architecture can be trained on, or used for any channel matrix whose height and width are multiples of 16, since the feature encoder has a total downsampling rate of 16 (or, of any size, which can be made a multiple of 16 by padding). This is another advantage of DeepCMC with respect to existing NN-based CSI compression techniques, which are all trained for a particular input size.

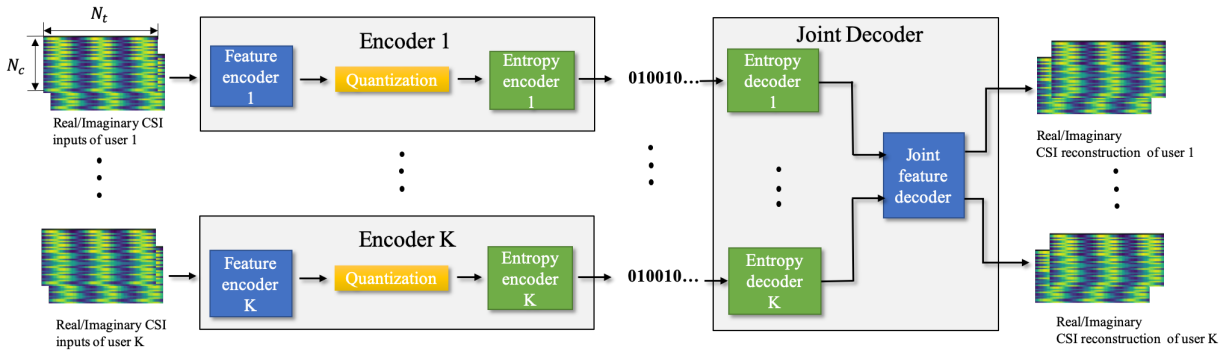

Fig. 4: The encoder/decoder architecture of DeepCMC for multiple-user scenario.

IV. Distributed DeepCMC

In a multi-user FDD massive MIMO scenario with K users, each user needs to compress and feedback its downlink CSI to the BS, separately. However, if the users are located close to each other, we expect their CSI matrices to be correlated as they share some common multipath components. Even though the compression is carried out separately at the the users, they can benefit from the correlation among their CSI matrices to achieve a better trade-off between the compression rate and the reconstruction MSE if the BS jointly reconstructs the CSI of multiple users from the received feedback messages. This is motivated by the information theoretic results on distributed lossy compression of correlated sources [23]. To this end, we propose a distributed DeepCMC NN architecture in which a joint feature decoder is used to simultaneously reconstruct the CSI matrix for several users at the BS.

Fig. 4 provides the overall block diagram of our proposed distributed DeepCMC architecture. According to this figure, a K user distributed DeepCMC architecture consists of K separate encoder branches each consisting of a feature encoder, quantization and entropy encoder blocks to compress the downlink CSI from users to K bitstreams. The feature encoder, quantization and entropy encoder block architectures are the same as described for the single user DeepCMC architecture. At the joint decoder, the bitstreams go through K separate entropy decoders with the same architecture as described in the previous section. The output of the entropy decoders are input to the joint feature decoder.

To design the joint feature decoder block, consider downlink CSI matrices of two nearby users denoted by $\mathbf{H}^1 = [\mathbf{h}_1^1, \mathbf{h}_2^1, \ldots, \mathbf{h}_{N_c}^1]^T$ and $\mathbf{H}^2 = [\mathbf{h}_1^2, \mathbf{h}_2^2, \ldots, \mathbf{h}_{N_c}^2]^T$. According to (2), $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$ can be written as the summation of multipath components. Note that, if the two users are located

close to each other, the components impinging from scatterers located far away from them appear with similar angle of arrival, gain, and delay values in their CSI matrices. Hence, $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$ share similar components coming from far scatters. This motivated us to use a summation-based joint feature decoder as depicted in Fig. 5, for $K = 2$. According to Fig. 5, the input from each entropy decoder is first processed separately by several convolutional layers, the structure of which is the same as feature decoder for single user DeepCMC excluding the last layer. Then the output is fed to two convolutional layers with kernel sizes of 9×9 to generate two different outputs, one of which contributes to the reconstruction of the CSI of user 1, while the other to the CSI of user 2. The CSI reconstruction of each user is obtained by the element-wise summation of two signals. This way, the joint decoder combines shared components from far away scatterers by summing them with appropriate combining kernels, thereby improving the reconstruction quality through diversity. The NN learns the optimal combining kernels of the last layer through training. Note that the architecture presented in Fig. 5 for the joint feature decoder can be easily generalized to any arbitrary value of K.

Our loss function for the distributed DeepCMC is given as follows:

$$L(\mathbf{\Theta}_{en}^{1:K}, \mathbf{\Theta}_{de}, \mathbf{\Theta}_{p}^{1:K}) = \mathbb{E}_{\mathbf{H}^{1:K}, \Delta\mathbf{M}^{1:K}} \Bigg(- \frac{1}{N_c N_t} \sum_{k=1}^{K} \log P^k(f_{\mathrm{f-en}}^k(\mathbf{H}^k, \mathbf{\Theta}_{en}^k) + \Delta\mathbf{M}^k, \mathbf{\Theta}_p^k) + \sum_{k=1}^{K} \lambda_k \mathrm{MSE}\Big(f_{\mathrm{f-jde}}\big(f_{\mathrm{f-en}}^1(\mathbf{H}^1, \mathbf{\Theta}_{en}^1) + \Delta\mathbf{M}^1, ..., f_{\mathrm{f-en}}^K(\mathbf{H}^K, \mathbf{\Theta}_{en}^K) + \Delta\mathbf{M}^K, \mathbf{\Theta}_{de}\big)[k], \mathbf{H}^k \Big) \Bigg), \tag{4}$$

where the superscript k specifies the corresponding user and we denote the sequence $\mathbf{X}^i, \mathbf{X}^{i+1}, \ldots, \mathbf{X}^j$ shortly by $\mathbf{X}^{i:j}$. In particular, $f_{\mathrm{f-en}}^k(\cdot, \mathbf{\Theta}_{en}^k)$ denotes the feature encoder at user k, parameterized by set $\mathbf{\Theta}_{en}^k$, $\Delta\mathbf{M}^k$ is the quantization noise vector with i.i.d elements from $U[0, 1]$ that is added to the feature encoder output to replace the quantization during training, and $P^k(\cdot, \mathbf{\Theta}_p^k)$ denotes the probability density function parameterized by $\mathbf{\Theta}_p^k$. The joint feature decoder is denoted by $f_{\mathrm{f-jde}}\big(\cdot, \mathbf{\Theta}_{de}\big)$, parameterized by $\mathbf{\Theta}_{de}$. We note that the joint feature decoder takes in all the outputs from K entropy decoders, and outputs the CSI reconstruction of all the K users. Hence, we denote by $f_{\mathrm{f-jde}}\big(\cdot, \mathbf{\Theta}_{de}\big)[k]$ the CSI reconstruction of user k output by the joint feature decoder. The expectation is taken over the training set of channel matrices and the quantization noise vectors. By minimizing this loss function, the sum entropy of the feedback data from all the K users (total overhead), and the weighted MSE of the reconstructed channel gain matrices

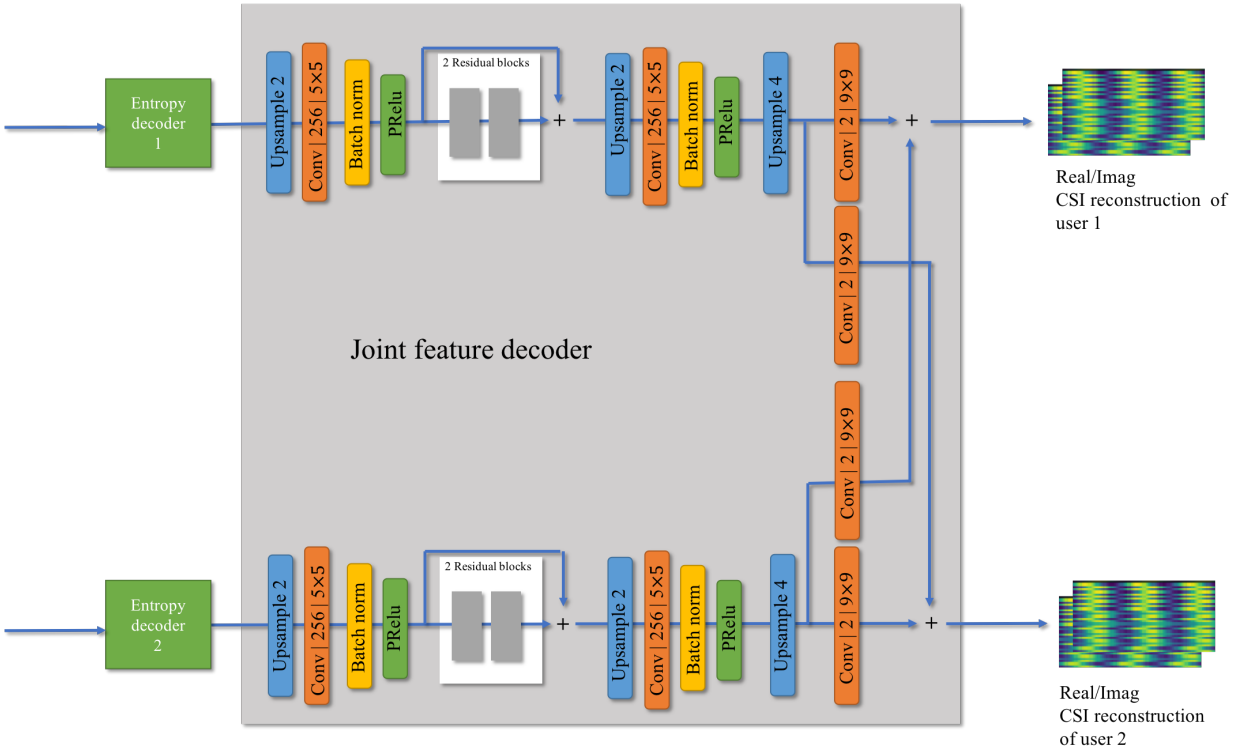

Fig. 5: Joint feature decoder architecture.

are jointly minimized. Similarly to the single user case, λ_k governs the trade-off between the feedback rate and the reconstruction quality. A larger λ_k results in a better reconstruction of channel matrix for user k but at an increased feedback overhead. Note that non-identical values of λ_1, .., λ_K allows heterogeneous reconstruction quality of the channel matrices.

V. Simulations

We use the COST 2100 channel model [30] to generate sample channel matrices for training and testing. We consider the indoor picocellular scenario at 5.3 GHz, where the BS is equipped with a ULA of dipole antennas positioned at the center of a $20\text{m} \times 20\text{m}$ square within which users are placed. We train our models on datasets of 80000 and test on 20000 CSI realizations generated by the COST 2100 model. Each CSI realization considers a random scattering environment following the default settings in [30]. We use the tensorflow compression library at [31] for DeepCMC implementation.

TABLE I: Performance comparison between DeepCMC and CSINet schemes for the single-user scenario in terms of NMSE and cosine correlation for similar bit rate values (The user is randomly placed, and $N_c = 256, N_t = 32$).

Methods	λ	Bit rate	Entropy	NMSE (dB)	ρ
DeepCMC	10^4	0.006068	0.003853	-4.12	0.8401
	5×10^4	0.01353	0.01152	-7.31	0.9337
	10^5	0.02266	0.02105	-9.08	0.9555
	5×10^5	0.05353	0.05478	-11.83	0.9732
	10^6	0.07658	0.07488	-12.45	0.9770
	5×10^6	0.1526	0.1509	-13.57	0.9808
CSINet	NA	0.015625	NA	-1.31	0.6903
	NA	0.03125	NA	-2.90	0.7806
	NA	0.0625	NA	-5.33	0.8856
	NA	0.1563	NA	-7.04	0.9314

We first present the performance of a single-user DeepCMC architecture in different scenarios in Subsection V.A, and then provide performance results for distributed DeepCMC in Subsection V.B. We use the normalized MSE (NMSE) and cosine correlation as the performance measures. These measures are defined as follows:

$$\mathrm{NMSE} \triangleq \mathbb{E}\left\{\frac{\|\mathbf{H}-\hat{\mathbf{H}}\|_2^2}{\|\mathbf{H}\|_2^2}\right\}, \tag{5}$$

and

$$\rho \triangleq \mathbb{E}\left\{\frac{1}{N_c}\sum_{n=1}^{N_c}\frac{|\hat{\mathbf{h}}_n^H\mathbf{h}_n|}{\|\hat{\mathbf{h}}_n\|\|\mathbf{h}_n\|}\right\}. \tag{6}$$

A. DeepCMC for a Single User

1) Bit rate-NMSE trade-off: We first compare the performance of our DeepCMC scheme with CSINet for the single-user scenario. We assume the user is placed uniformly at random within a $20\mathrm{m} \times 20\mathrm{m}$ square area where the BS is positioned at the center point $(10\mathrm{m}, 10\mathrm{m})$. We set $N_c = 256$ and $N_t = 32$. Table I provides the corresponding results tested on 20000 CSI realizations with the same $N_c = 256$ and $N_t = 32$. In this table, We train our DeepCMC

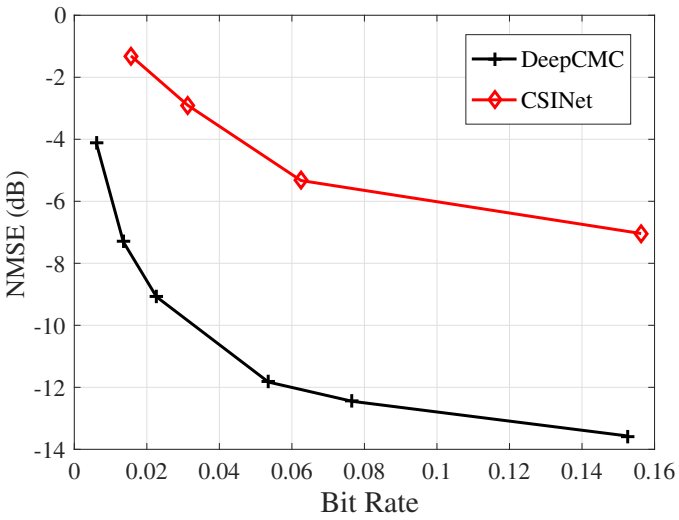

Fig. 6: Bit rate-NMSE trade-off of DeepCMC vs. CSINet, $N_c = 256$, $N_t = 32$.

architecture with different λ values, which governs the trade-off between the compression rate and the reconstruction quality. We evaluate both the average entropy of the quantized outputs of the feature encoder and the average number of actual bits to transmit back to the BS. The actual number of bits includes the length of the bit stream generated by the entropy encoder plus 16 additional bits to transmit the value of lambda to the BS so that the BS can use the NN trained with the proper λ value to reconstruct the CSI. Hence, the actual bit rate will further reduce if the BS and the user agree on a fixed λ value throughout their operation. Both the average entropy and the number of bits are normalized by N_cN_t, the CSI matrix dimension, to represent the average bit rate per CSI value. According to the results in Table I, the actual bit rate closely approximates the entropy of the quantized feature encoder outputs. On the other hand, CSINet provides a feature vector of 32-bit float values. The length of this vector, denoted by m, determines the compression ratio, and hence the bit rate for CSINet. We train CSINet for $m = 8, 16, 32, 80$, which correspond to bit rates of 0.01562, 0.03125, 0.0625, and 0.1563, respectively.

The bit rate-NMSE trade-offs achieved by DeepCMC and CSINet are plotted in Fig. 6. As it can be observed from Table 1 and Fig. 6, DeepCMC provides significant improvement in the quality of the reconstructed CSI at the BS with respect to CSINet at all bit rate values. We remark here that, as reported in [5], CSINet itself provides $3 - 6$ dB improvement in NMSE

compared to previous MIMO CSI compression techniques based on sparsity of the channel gain matrix. However, the gains from DeepCMC are even more drastic, achieving remarkably good reconstruction of the channel gain matrix with NMSE of -13 dB and ρ equal to 0.98 at a bit rate lower than 0.16 bits per channel dimension. These results show that DeepCMC outperforms CSINet 4 to 6 dB in NMSE for the range of compression rates considered here. For example, for a target value of $\mathrm{NMSE} = -5$ dB, DeepCMC can provide more than 5 times reduction in the number of bits that must be fed back from the user to the BS. We further observe from the rate-distortion curves in Fig. 6 that the NMSE of DeepCMC drops quite rapidly with bit rate, while CSINet shows a smoother reduction slope. This implies that DeepCMC better exploits the limited number of bits to capture the most essential information in the CSI data.

These improvements are not only due to our improved feature extraction architecture, but also incorporation of the quantization and entropy coding blocks in the training procedure which enables efficient compression of the quantizer output at rates very close to its entropy. The entropy coder can efficiently convert the quantizer output to bits by utilizing its probability distribution estimated during training. Our experiments also reveal that adding the shortcut connections across two residual blocks at the decoder and choosing PRelu (in comparison with Relu and Leaky Relu) as the activation function improves performance of DeepCMC.

2) Stationary user: For a user fixed at a certain position from the BS, we can use COST2100 to generate a dataset for that specific position and train our DeepCMC network with it. This could be the case where a wireless user is stationary (e.g., desktop PC, smart home appliances, etc. in the indoor scattering scenario) and will significantly improve the performance as there is less information in the CSI matrix of a fixed user to compress. Note that although the user is fixed in this scenario, the scattering environment, the corresponding multi-path signal components and consequently the CSI matrices vary.

To study the performance in this scenario, we train and test both DeepCMC and CSINet for a user fixed at $(5\mathrm{m}, 5\mathrm{m})$. Table II provides the corresponding results for $N_c = 256$ and $N_t = 32$. According to Table II, both DeepCMC and CSINet benefit significantly from the fixed user location with up to 25 dB reduction in NMSE for DeepCMC and 8 dB for CSINet in comparison with the results in Table I. The gains from DeepCMC compared to CSINet are even more drastic for a fixed user, achieving almost perfect reconstruction of the channel gain matrix with NMSE of -40 dB and ρ approximately equal to 1 at a bit rate lower than 0.15 bits per channel dimension. The results presented in Table II show that DeepCMC outperforms CSINet

TABLE II: Performance of DeepCMC and CSINet schemes for the single-user scenario in terms of NMSE and cosine correlation for similar compression rate values (bits per channel dimension) (The user is placed at fixed location, and $N_c = 256, N_t = 32$).

Methods	λ	Bits rate	Entropy	NMSE (dB)	ρ
DeepCMC	100	0.01277	0.01069	-15.02	0.9903
	1000	0.03428	0.02868	-23.39	0.998
	5000	0.05743	0.05377	-28.79	0.9995
	10000	0.06864	0.07709	-31.65	0.9998
	50000	0.1163	0.1079	-37.23	0.9999
	100000	0.1459	0.1411	-39.33	1
CSINet	NA	0.015625	NA	-9.43	0.9694
	NA	0.03125	NA	-10.26	0.9732
	NA	0.0625	NA	-11.74	0.9785
	NA	0.125	NA	-12.67	0.9837

TABLE III: Performance of DeepCMC (trained with $\lambda = 10^5$) when users are located with different distances to the BS, $N_c = 256, N_t = 32$

Distance	Bit rate	Entropy	NMSE (dB)	ρ
$2.5\mathrm{m}$	0.02938	0.02772	-13.33	0.9835
$5\mathrm{m}$	0.0213	0.01964	-11.01	0.9734
$7.5\mathrm{m}$	0.0192	0.01753	-8.94	0.9586
$10\mathrm{m}$	0.01944	0.01777	-3.56	0.8498
random	0.02266	0.02105	-9.08	0.9555

6 to 20 dB in NMSE for the range of compression rates considered here. For example, for a target value of $\mathrm{NMSE} = -15$ dB, DeepCMC can provide 10 times reduction in the number of bits that must be fed back from the user to the BS.

3) User position uncertainty: For the general scenario where the users may move, we train DeepCMC with dataset entries generated for users randomly placed in a training area. We have so far considered a $20\mathrm{m} \times 20\mathrm{m}$ square training area with the BS positioned at the center at $(10\mathrm{m}, 10\mathrm{m})$. We here study the performance of our DeepCMC network trained for the $20\mathrm{m} \times 20\mathrm{m}$

square area for users placed on circles at different distances, in particular, $2.5\mathrm{m}$, $5\mathrm{m}$, $7.5\mathrm{m}$, $10\mathrm{m}$ around the BS. We summarized the performance of DeepCMC, trained with $\lambda = 10^5$, with regards to the distance between the user and the BS in Table III. The last row shows the performance when the user is randomly located within the square. Although the reconstruction performance degrades as the user moves further away from the BS, it still remains acceptable ($\mathrm{NMSE} < -3\mathrm{dBs}$) as long as the user stays within the training area. The NMSE for DeepCMC is smaller when the user is closer to the BS at a slightly larger bit rate.

4) Performance on wide band MIMO systems: In practical MIMO scenarios, the bandwidth and consequently number of subcarriers N_c may change from system to system or over time due to time-varying resource allocation. Hence, it is desirable for any CSI feedback scheme to maintain an acceptable performance as the number of subcarriers changes, so that the users will not need to store different NN parameters trained for different bandwidths. Unlike the previous works, which include dense layers in their NN architectures, DeepCMC, being fully convolutional, is applicable to scenarios with different N_c values.

We design experiments to evaluate the performance of DeepCMC when trained on $N_c = 256$ but tested on $N_c = 128, 160, 192, 224, 256, 512, 1024$. We summarized the performance of DeepCMC, trained with $\lambda = 10^5$, in Table IV. We also present the bit rate-NMSE trade-off in Fig. 7, which is obtained by testing the DeepCMC (trained with different λ values) on different values of N_c. According to Table IV and Fig. 7, the DeepCMC convolution kernels once trained for $N_c = 256$, work sufficiently well both on smaller and larger values of N_c in a wide range of three octaves ($\frac{1024}{128} = 8$). This is very desirable as it makes our proposed DeepCMC architecture applicable to wide band massive MIMO systems. Also according to Fig. 7, CSI matrices for wide band MIMO scenarios seem to be more compressible as larger N_c values result in lower bit rate and better NMSE.

Note that, although a DeepCMC network trained on a dataset with $N_c = 256$ provides very good rate-distortion curves for $N_c = 128$ and 1024 according to Fig. 7, we are interested to compare its performance with networks trained specifically on $N_c = 256$ and $N_c = 1024$. The corresponding comparison results are provided in Fig. 8. According to this figure, although networks trained and tested on the same N_c values provide better performance, the performance gap is small if N_c is different for train and test. This shows that utilizing DeepCMC, the UE can use the kernels optimized for a specific N_c value to compress the CSI for a wider range of bandwidths with negligible performance loss.

TABLE IV: Performance of DeepCMC (trained with $\lambda = 10^5$) for different number of subcarriers in the test channel with $N_t = 32$

N_c	Bit rate	Entropy	NMSE (dB)	ρ
128	0.02493	0.02301	-8.14	0.9469
160	0.02388	0.02217	-8.29	0.9474
192	0.02318	0.02163	-8.42	0.9478
224	0.02269	0.02124	-8.51	0.9480
256	0.02232	0.02094	-8.60	0.9482
512	0.021	0.0199	-9.01	0.9490
1024	0.02035	0.01938	-9.28	0.9496

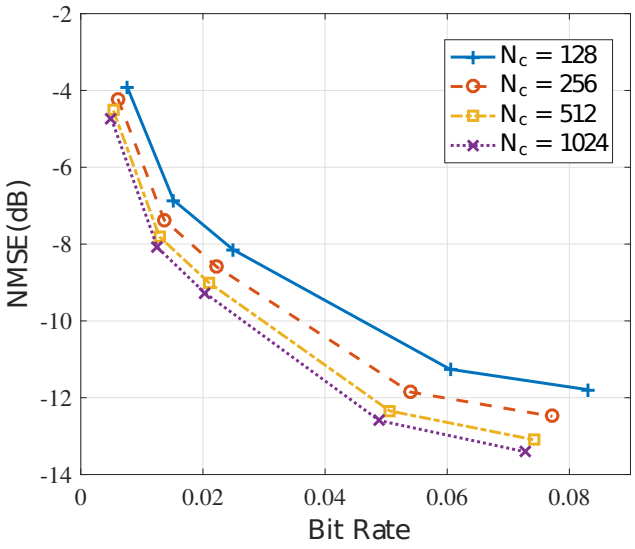

Fig. 7: Bit rate-NMSE trade-off for different number of subcarriers for a DeepCMC network trained with $N_c = 256$, $N_t = 32$.

B. Distributed DeepCMC for Two Nearby Users

In this subsection, we study the performance of our proposed distributed DeepCMC architecture for two users. To this end we place two users around the BS such that they are 30cm and 60cm apart from each other, respectively. We place the users at $(5\text{m}, 5\text{m})$ and $(4.7\text{m}, 5\text{m})$ for the 30cm distance case and at $(5\text{m}, 5\text{m})$ and $(4.4\text{m}, 5\text{m})$ for the 60cm case and use COST2100 to simultaneously generate CSI datasets for the two users with $N_t = 32$ and $N_c = 256$. We train

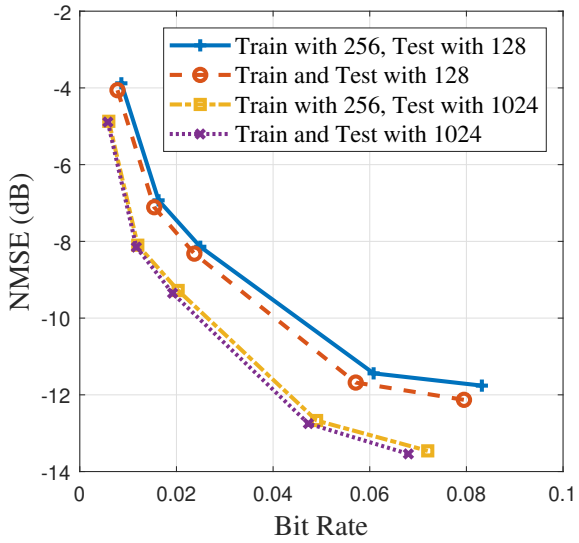

Fig. 8: DeepCMC performance when N_c is different for train and test, $N_t = 32$.

with 80000 CSI realizations, and test over 20000 independent realizations. Other simulation parameters are the same as those in Subsection V-A.

We compare two scenarios. In the first scenario, we encode and decode the CSI for the two users using two independent DeepCMC networks trained separately for the two users. This approach does not benefit from the common structure and correlations shared by the users. In the second scenario, we train and use distributed DeepCMC to encode separately, but decode jointly. Fig. 9 compares the bit rate-NMSE curves for the two scenarios for two users placed 30cm apart. According to Fig. 9, the bit rate-NMSE curves for distributed DeepCMC always lie below the single user DeepCMC for both users and the performance improvement by distributed DeepCMC becomes more significant as the bit rate increases. This is in line with the intuition; at very low bit rates, each user compresses and sends the most important components of its CSI matrix, which are not necessarily useful for the other user. As more rate becomes available, they start transmitting more common structures, which can be useful for both users. This is also in line with information theoretic results on distributed data compression [23]. Using distributed DeepCMC, a 15.3% reduction in the required bit rate of the first user is observed at a reconstruction NMSE of -35.5 dB.

In Fig. 10, we plot the sum rate against the sum NMSE for two users placed 30 and 60 cm apart. According to Fig. 10, the improvement by distributed DeepCMC is more significant as

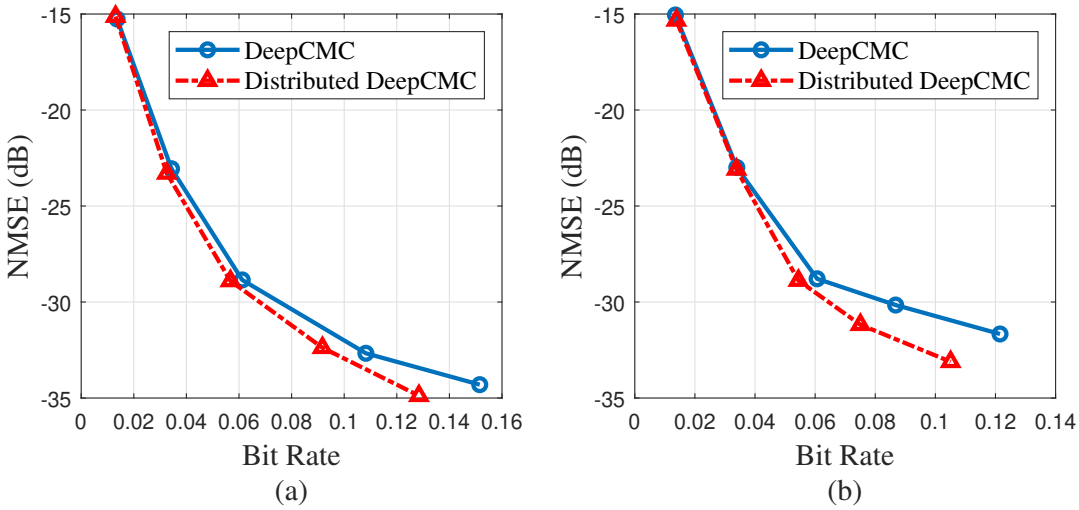

Fig. 9: Bit rate-NMSE curves of DeepCMC and distributed DeepCMC for two users located 30cm apart. (a) user 1 (b) user 2.

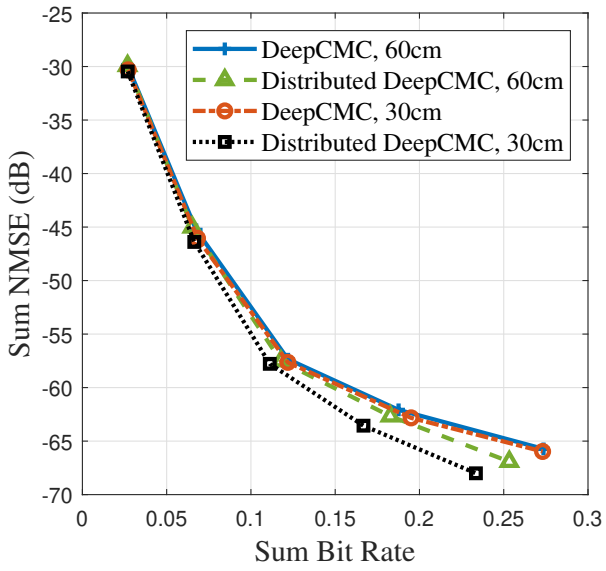

Fig. 10: Bit rate-NMSE comparison for DeepCMC and distributed DeepCMC where the distance between two users is 30 and 60 cm.

the distance between the users decreases. This is expected because the CSI matrices for closer users share more similar multipath components.

VI. Conclusion

In this paper, we proposed a convolutional DL architecture, called DeepCMC, for efficient compression of CSI matrices to reduce the significant CSI feedback overhead in massive MIMO systems. DeepCMC is composed of fully convolutional layers followed by quantization and entropy coding blocks, and outperforms state of the art DL-based CSI compression techniques, providing drastic improvements in CSI reconstruction quality at even extremely low feedback rates. We also proposed a distributed version of DeepCMC for a multi-user MIMO scenario such that different users compress their CSI matrices in a distributed manner, which are reconstructed jointly at the BS. Distributed DeepCMC not only utilizes the inherent CSI structures of a single MIMO user for compression, but also benefits the channel correlations among nearby MIMO users to further improve the performance in comparison with DeepCMC. We showed that distributed deepCMC can provide further reduction in the feedback overhead, particularly for nearby users, and at higher bit rates.

References

[1] Q. Yang, M. B. Mashhadi, and D. Gündüz, "Deep convolutional compression for massive MIMO CSI feedback," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 13-16 Oct 2019, pp. 1–6.

[2] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, November 2010.

[3] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 40–60, Jan 2013.

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

[5] P. Kuo, H. T. Kung, and P. Ting, "Compressive sensing based channel feedback protocols for spatially-correlated massive antenna arrays," in *IEEE Wireless Commun. and Netw. Conf. (WCNC)*, April 2012, pp. 492–497.

[6] X. Rao and V. K. N. Lau, "Distributed compressive CSIT estimation and feedback for FDD multi-user massive MIMO systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3261–3271, June 2014.

[7] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. on Cognitive Communications and Networking*, vol. PP, no. 99, 2017.

[8] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2184–2199, Oct 2019.

[9] M. B. Mashhadi and D. Gündüz, "Deep learning for massive MIMO channel state acquisition and feedback," *arXiv:2002.06945 [cs.IT]*, Feb. 2020.

[10] Z. Liu, L. Zhang, and Z. Ding, "Overcoming the channel estimation barrier in massive MIMO communication systems," *arXiv:1912.10573 [cs.IT]*, Dec. 2019.

[11] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive MIMO CSI feedback," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 748–751, 2018.

[12] T. Wang, C. Wen, S. Jin, and G. Y. Li, "Deep learning-based CSI feedback approach for time-varying massive MIMO channels," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 416–419, April 2019.

[13] C. Lu, W. Xu, H. Shen, J. Zhu, and K. Wang, "MIMO channel information feedback using deep recurrent network," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 188–191, Jan 2019.

[14] Z. Liu, L. Zhang, and Z. Ding, "Exploiting bi-directional channel reciprocity in deep learning for low rate massive MIMO CSI feedback," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 889–892, June 2019.

[15] Y. Liao, H. Yao, Y. Hua, and C. Li, "CSI feedback based on deep learning for massive MIMO systems," *IEEE Access*, vol. 7, pp. 86 810–86 820, 2019.

[16] M. B. Mashhadi, Q. Yang, and D. Gündüz, "CNN-based analog CSI feedback in FDD MIMO-OFDM systems," in *IEEE International Conference on acoustics, Speech and Signal Processing (ICASSP 2020)*, May 2020.

[17] Y. Jang, G. Kong, M. Jung, S. Choi, and I. Kim, "Deep autoencoder based CSI feedback with feedback errors and feedback delay in FDD massive MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 833–836, June 2019.

[18] J. Guo, C. Wen, S. Jin, and G. Y. Li, "Convolutional neural network based multiple-rate compressive sensing for massive MIMO CSI feedback: Design, simulation, and analysis," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020.

[19] Z. Lu, J. Wang, and J. Song, "Multi-resolution CSI feedback with deep learning in massive MIMO system," *arXiv:1910.14322 [cs.IT]*, Oct. 2019.

[20] C. Lu, W. Xu, S. Jin, and K. Wang, "Bit-level optimized neural network for multi-antenna channel quantization," *IEEE Wireless Communications Letters*, vol. 9, no. 1, pp. 87–90, Jan 2020.

[21] Z. Liu, L. Zhang, and Z. Ding, "An efficient deep learning framework for low rate massive MIMO CSI reporting," *arXiv:1912.10608 [cs.IT]*, Dec. 2019.

[22] N. Shlezinger and Y. C. Eldar, "Deep task-based quantization," *arXiv: 1908.06845[eess.SP]*, Aug. 2019.

[23] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, 2011, p. 294.

[24] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, January 1976.

[25] T. A. Courtade and T. Weissman, "Multiterminal source coding under logarithmic loss," in *2012 IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 761–765.

[26] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard," *IEEE Trans. Circuits and Syst. for Video Tech.*, vol. 13, no. 7, pp. 620–636, 2003.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Int'l Conf. Comp. vision and pattern recognition (CVPR)*, Las Vegas, NV, Jun. 2016, pp. 770–778.

[28] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional probability models for deep image compression," in *Proc. IEEE Int'l Conf. Comp. vision and pattern recognition (CVPR)*, Salt Lake City, UT, Jun 2018, pp. 4394–4402.

[29] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. IEEE Int'l Conf. on Learning Representations (ICLR)*, Vancouver, Canada, April 2018.

[30] L. Liu, C. Oestges, J. Poutanen, K. Haneda, P. Vainikainen, F. Quitin, F. Tufvesson, and P. D. Doncker, "The COST 2100 MIMO channel model," *IEEE Wireless Commun.*, vol. 19, no. 6, pp. 92–99, December 2012.

[31] https://tensorflow.github.io/compression/.