Xing and Hu
EURASIP Journal on Advances in Signal Processing (2022) 2022:47
https://doi.org/10.1186/s13634-022-00879-y

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Convolutional neural network and clustering-based codebook design method for massive MIMO systems

Jing Xing* and Die Hu

*Correspondence: xingjing@fudan.edu.cn

Department of Communication Science and Engineering, Fudan University, Shanghai, China

Abstract

In this paper, we propose a convolutional neural network (CNN) and clustering-based codebook design method. Specifically, we train two different CNNs, i.e., CNN1 and CNN2, to compress the channel state information (CSI) matrices into the channel vectors and recover the channel vectors back into the CSI matrices, respectively. After that, the clustering algorithm clusters the output of CNN1, i.e., the channel vectors into several clusters and outputs a centroid for each cluster. The sum distance between each centroid and the channel vectors in the corresponding cluster is the smallest, which can lead to the maximum sum rate of massive MIMO codebook design. Then, the centroids are recovered into matrices by CNN2. The output of CNN2 is our proposed codebook for massive multiple-input multiple-output (MIMO) systems. In the simulation, we compare the performance of different clustering algorithms. We also compare the proposed codebook with the traditional discrete Fourier transform (DFT) codebook. Simulation results show the superiority of the proposed algorithm.

Keywords: Convolutional neural network (CNN), Codebook design, Clustering

1 Introduction

Recently, millimeter wave (mmWave) cellular system is developing rapidly because of the large available bandwidth at mmWave frequencies. Large numbers of antennas are employed at both the base station (BS) and the user equipment (UE), which is called the massive multiple-input multiple-output (MIMO) systems. In order to realize the potential gains of massive MIMO and obtain the beamforming gains and the spatial multiplexing gains, the channel state information (CSI) matrix needs to be known at transmitter including the BS and the UE, and the accuracy of CSI is crucial for the precoding performance of the massive MIMO systems [1]. The CSI feedback overhead is increasing with the number of antennas, which makes the transmission of the CSI matrix difficult, especially in the frequency division duplex (FDD) system [2, 3].

For the uplink, the UE sends the pilot signal to the BS so that the BS can estimate the CSI accurately. In the time division duplex (TDD) systems, the uplink and downlink use the same frequency to transmit the signal. Therefore, the BS can obtain the downlink CSI through the reciprocity between uplink and downlink. As for the FDD

systems, the UE needs to estimate the downlink CSI matrix and feeds it back to the BS, which is impossible for the UE because of the high dimension of the CSI matrix.

Therefore, many researchers are dedicated to reducing the overhead of CSI feedback. Deep learning-based methods have recently made outstanding strides in CSI matrix feedback. In [4], a deep neural network (DNN) is proposed to obtain the mapping of the downlink CSI matrix and the low-rank matrix. The CSI matrix is recovered according to the low-rank matrix at the BS. In [5], an encoder–decoder structure-based convolutional neural network (CNN) is proposed to compress and recover the CSI matrix at the UE and the BS. In [6], a CNN-based feedback scheme called AnalogDeepCMC is proposed to map the downlink channel to uplink CSI and to reconstruct the downlink channel.

The above methods all reduce the CSI feedback bits to a certain extent, but the number of feedback bits is still not negligible. In order to further reduce the feedback overhead, a lot of research has been devoted to design the limited feedback-based codebook that is shared between the transmitter and the receiver [7]. The UE only needs to select the best codeword from the codebook and feeds back the index of the corresponding codeword to the BS. The design of the codebook is extremely important because it directly determines the efficiency of precoding. The discrete Fourier transform (DFT)-based codebook is proposed in [8], in which the precoding vector is actually the column vector of the DFT matrix. In [9], an improved algorithm based on the DFT codebook is proposed to ensure the orthogonality between each codeword, thereby improving the performance of the DFT-based codebook. The DFT-based codebook is considered as a compact and efficient codebook design, but it is only suitable for uniform linear arrays (ULA). In fact, the codebook design should be able to accommodate complex wireless channels and different antenna arrays. In codebook-based algorithms, transmitter selects the most appropriate codeword to transmit the signal according to the current channel, which can be regarded as a classification matching problem. Clustering algorithm is used for channel classification because of its ability to process massive MIMO channel. In [10], the clustering algorithm processed a large amount of channel data and outputs the corresponding centroids for codebook design.

Inspired by the above articles, we propose a novel codebook design method combining deep learning and clustering algorithm, which can take advantage of both state-of-the-art technologies. We first train two different CNNs to compress the CSI matrix into channel vector and to recover the channel vector back into matrix, called CNN1 and CNN2, respectively. We use the well-trained CNN1 to compress a large number of channel matrices and obtain the channel vectors. Subsequently, the clustering algorithm is applied to cluster the channel vectors into $K = 2^B$ centroids, where B is the number of feedback bits. The sum of the distance between the centroid and the channel vectors in the corresponding cluster is minimal. The well-trained CNN2 recovers the K centroids into the matrices, and the output of CNN2 is our proposed codebook.

The remaining content of this article is structured as follows: Sect. 2 introduces the system model and the traditional DFT-based codebook. Section 3 describes the proposed CNN and clustering-based codebook. The simulation results and analysis are given in Sect. 4, and Sect. 5 provides the conclusions.

2 System model and traditional DFT-based codebook method

In this paper, we consider a downlink single-cell massive MIMO system with N_t antennas at the BS communicating with a single-antenna user. The system is operated in orthogonal frequency division multiplexing (OFDM) over N_c subcarriers. The received signal at the nth subcarrier can be expressed as:

$$y_n = \mathbf{h}_n^H \cdot \mathbf{w}_n \cdot s_n + z_n \tag{1}$$

where $n = 1, 2, \ldots, N_c$, $\mathbf{h}_n \in \mathbb{C}^{N_t \times 1}$ denotes the channel vector, $(\cdot)^H$ represents Hermitian transpose, $\mathbf{w}_n \in \mathbb{C}^{N_t \times 1}$ is the precoding vector and $\|\mathbf{w}_n\|^2 = 1$, $\|\cdot\|^2$ represents the Frobenius norm, $s_n \in \mathbb{C}$ is the transmitted signal, $z_n \in \mathbb{C}$ is the complex Gaussian noise with zero mean and variance σ_z^2.

In massive MIMO systems, the accuracy of channel vector $\mathbf{h}_n$ is critical for the precoding [11]. In FDD systems, the UE needs to feed the estimated $\mathbf{h}_n$ to the BS directly to calculate the precoding vector $\mathbf{w}_n$ of the downlink [12]. Taking into account all subcarriers, there are $N_t N_c$ elements to feed back in this situation, which is enormous in massive MIMO systems. Hence, codebook-based precoding schemes became a more practical method thanks to its minimal feedback overhead [1].

DFT-based codebook is considered as an effective design for spatially correlated channels. In DFT-based precoding schemes, the precoding vector $\mathbf{w}_n$ is chosen from the codebook $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \ldots, \mathbf{c}_K]$, where $K = 2^B$ is the number of codewords that the codebook contains, B is the number of feedback bits per subcarrier, $\mathbf{c}_i$ can be expressed as [9]:

$$\mathbf{c}_i = \frac{1}{\sqrt{N_t}} \left[1, e^{-j2\pi \cdot 1 \cdot i}, e^{-j2\pi \cdot 2 \cdot i}, \ldots, e^{-j2\pi \cdot (N_t - 1) \cdot i} \right]^T \tag{2}$$

where $i = 0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{N-1}{K}$, and $(\cdot)^T$ represents transpose.

DFT-based codebook design is independent of $\mathbf{h}_n$, which makes it only applicable to highly correlated channels [13]. Besides, there are N_c subcarriers, which means that the number of total feedback bits is $N_c B$. To solve the above problem, we propose a novel codebook design to make the codebook more suitable for different CSI while also greatly reducing the amount of feedback bits.

3 Proposed method based on CNN and clustering algorithm

In this section, we introduce the main idea of the proposed codebook design in detail. First, we convert the codebook design into an optimization problem. Then, we train the CNNs and exploit clustering algorithms to address the optimization problem.

3.1 Problem formulation

Rewrite (1) in a vector form as:

$$\mathbf{y} = \mathbf{HWs} + \mathbf{z} \tag{3}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_{N_c}]^T$, $\mathbf{s} = [s_1, s_2, \ldots, s_{N_c}]^T$, and $\mathbf{z} = [z_1, z_2, \ldots, z_{N_c}]^T$ are all $N_c \times 1$ vectors, $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{N_c}]^H \in \mathbb{C}^{N_c \times N_t}$ is the CSI matrix, and $\mathbf{W} = \left[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{N_c}\right] \in \mathbb{C}^{N_t \times N_c}$ is the precoding matrix.

The achievable rate of the nth subcarrier can be expressed as [14]:

$$R_n = \log_2\left(1 + \gamma|\mathbf{h}_n^H \mathbf{w}_n|^2\right) \tag{4}$$

where γ is signal–noise ratio (SNR). Consequently, the total achievable rate can be obtained by summing up the achievable rate of all subcarriers as

$$\begin{aligned} R &= \sum_{n=1}^{N_c} \log_2\left(1 + \gamma\left|\mathbf{h}_n^H \mathbf{w}_n\right|^2\right) \\ &= \sum_{n=1}^{N_c} \log_2\left(1 + \gamma\left|[\mathbf{HW}]_{n,n}\right|^2\right) \end{aligned} \tag{5}$$

where $[\cdot]_{n,n}$ represents the element at nth row and nth column in the matrix.

From (5), it can be seen that the achievable rate is a function of precoding matrix $\mathbf{W}$, i.e.,

$$R = f(\mathbf{W}) \tag{6}$$

The main goal of precoding is to improve the achievable rate R by designing $\mathbf{W}$. To reduce the feedback overhead, we design $\mathbf{W}$ instead of $\mathbf{w}_n$ in this paper, i.e., we chose $\mathbf{W}$ from the codebook $\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \ldots, \mathbf{C}_K]$, where $\mathbf{C}_i, i = 1, 2, \ldots, K$ is the $N_t \times N_c$ codeword shared at the UE and the BS. Figure 1 shows the block diagram of the codebook-based downlink precoding scheme. The UE estimates $\mathbf{H}$ and selects the codeword from the codebook $\mathbf{C}$ that can maximize the sum achievable rate, i.e.,

$$\mathbf{W} = \arg\max_{\mathbf{C}_i \in \mathbf{C}} \left(\sum_{n=1}^{N_c} \log_2\left(1 + \gamma\left|(\mathbf{HC}_i)_{n,n}\right|^2\right)\right) \tag{7}$$

Then, the UE feeds back the index of corresponding codeword to the BS. Compared with using the DFT-based codebook, our proposed scheme can reduce the number of feedback bits from $N_c B$ to B.

Now the problem remaining is how to design the codebook $\mathbf{C}$. In this paper, we design $\mathbf{C}$ to maximize the average achievable rate, i.e.,

$$\max_{\mathbf{C}} E[R] \tag{8}$$

where $E[\cdot]$ means the mathematical expectation.

Define the distance between $\mathbf{H}$ and $\mathbf{C}_i$ as

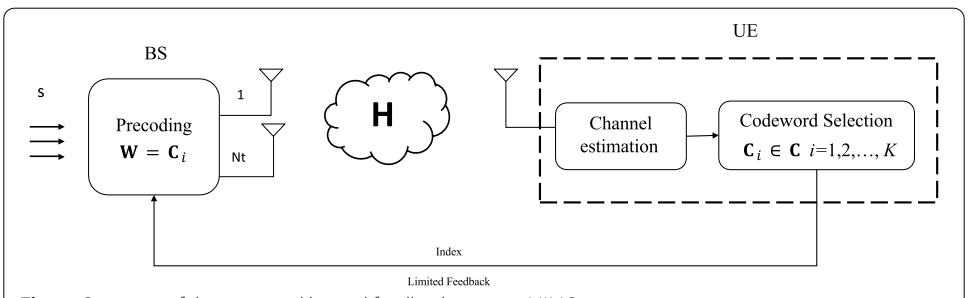

Fig. 1 Overview of the proposed limited feedback massive MIMO systems

$$d(\mathbf{H}, \mathbf{C}_i) = \frac{E\left[\sum_{n=1}^{N_c} ||\mathbf{h}_n - \mathbf{c}_{i,n}||^2\right]}{E\left[\sum_{n=1}^{N_c} ||\mathbf{h}_n||^2\right]} \tag{9}$$

where $\mathbf{c}_{i,n}$ is the nth column of $\mathbf{C}_i$. Through simulations, we observe that the smaller the $d(\mathbf{H}, \mathbf{C}_i)$ is, the higher the $E[R]$ is. Therefore, we transfer the optimization problem (8) to minimize the average distance, i.e.,

$$\max_{\mathbf{C}_i} E[R] \iff \min_{\mathbf{C}_i \in \mathbf{C}} E[d(\mathbf{H}, \mathbf{C}_i)] \tag{10}$$

In the next subsection, we will introduce our proposed codebook design based on the above optimization criteria.

3.2 Codebook design

In our proposed codebook design, two different CNNs, i.e., CNN1 and CNN2, are used to compress the CSI matrices and reconstruct the channel matrices. In the offline phase, we created the channel dataset under the COST 2100 channel model [15] as the input of CNN1, which is applied to extract the characteristics of numerous channel matrices and outputs the channel vectors. The purpose of CNN2 is to restore the channel vector to channel matrix which has the smallest distance from the CSI matrix. After the training procedure, we exploit the clustering algorithms to cluster the output of CNN1 and obtain several channel vectors, i.e., centroids, whose sum distance is minimum to the real channel vectors. As mentioned before, in order to maximize the total rate, we need to minimize the distance between the codeword and the real channel. Therefore, we use CNN2 to reconstruct the centroids and the output of CNN2 is the final codebook. The offline training and clustering is shown in Fig. 2a. The offline process is done at BS. The powerful processing capability of BS ensures the efficiency of the offline process.

In the actual online estimation, CNN1 at the UE first compresses the CSI matrix into a channel vector, which matches the nearest centroid. The UE feeds back the index of the centroid to the BS for selecting the corresponding codeword. Figure 2b shows the online codeword selection and feedback process.

In Sect. 3.2.1, we will introduce the structure and the training of CNN1 and CNN2. Section 3.2.2 introduces the clustering algorithm used in this paper.

3.2.1 CNNs' structure and training

Considering the sparsity of massive MIMO channels in the angular–delay domain, we first use 2D-DFT to convert the spatial frequency domain into the angular–delay domain, aiming to reduce the complexity and difficulty of feature extraction [16, 17]. Specifically, we obtain $\overline{\mathbf{H}}$ as

$$\overline{\mathbf{H}} = [\widetilde{\mathbf{H}}]_{[1:N_s],:} \tag{11}$$

where $[\mathbf{A}]_{[1:N_s],:}$ denotes the first N_s rows of the matrix $\mathbf{A}$ ($N_s < N_t$),

$$\widetilde{\mathbf{H}} = \mathbf{D}_t \mathbf{H} \mathbf{D}_a{}^H \tag{12}$$

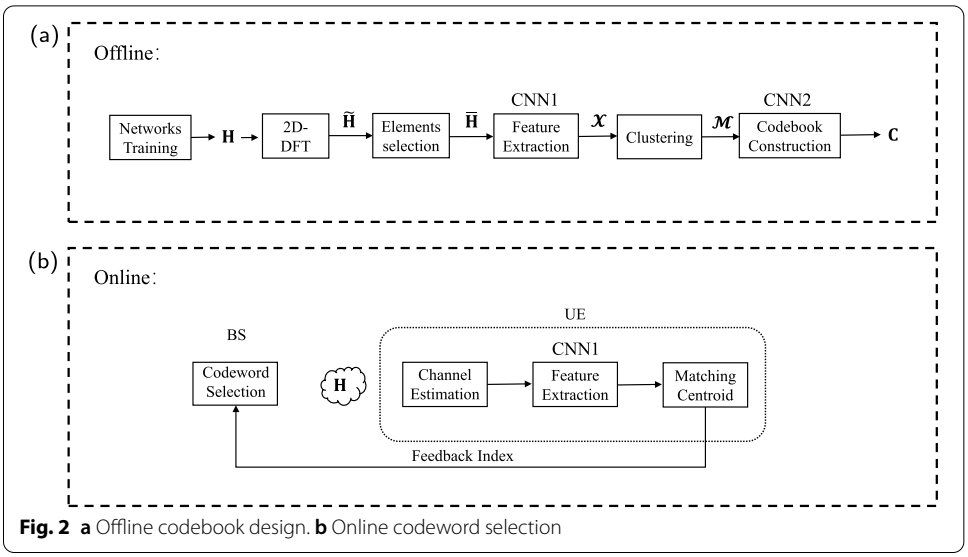

Fig. 2 **a** Offline codebook design. **b** Online codeword selection

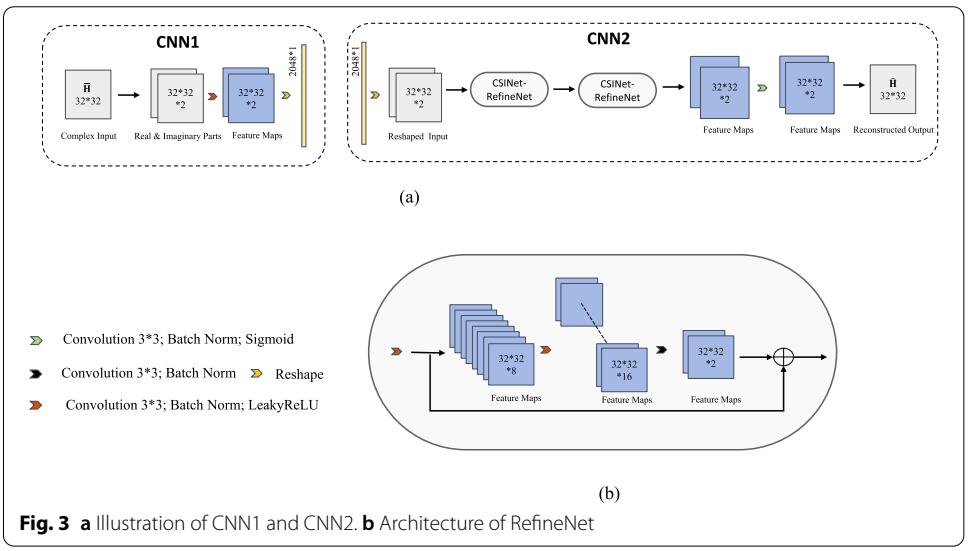

Fig. 3 **a** Illustration of CNN1 and CNN2. **b** Architecture of RefineNet

$\mathbf{D}_t$ and $\mathbf{D}_a$ are $N_c \times N_c$ and $N_t \times N_t$ DFT matrices, respectively. It can be found that most elements of $\widetilde{\mathbf{H}}$ are almost zero and only the first N_s rows have large components [5]. By choosing the first N_s rows of $\widetilde{\mathbf{H}}$, we can reduce the dimension of datasets and the training time.

We applied the encoder and decoder of CsiNet [16, 17] to build CNN1 and CNN2, respectively. The architectures are shown in Fig. 3a, in which the values $A_1 \times A_2 \times A_3$ represent the length, width, and the number of feature maps.

Since CNN can not handle complex number, the real and imaginary parts of $\overline{\mathbf{H}}$ are processed separately in the input layer. After the input layer, the convolution layer with batch normalization (BN) is used to extract channel feature and get the feature maps, where the size of convolution kernel is 3×3 and the activation function is LeakyReLU. Then, the output feature maps are reshaped into a 2048×1

vector $\mathbf{x}$, which is a real-valued channel vector. It should be pointed out that $\mathbf{x}$ is inputted to the full connection (FC) layer in CsiNet, whose dimension is further compressed to L under different compress rate r_c ($r_c = L/2\widetilde{N}_c N_t$). Here, different from CsiNet, we retain all the elements of $\mathbf{x}$ to ensure that the subsequent reconstructing can get more accurate results. The most important component of CNN1 is a convolution layer with a convolution kernel size of 3×3. The time complexity is $O\left(32^2 \times 3^2 \times 2 \times 2\right) = O(36864) \approx O(3.7 \times 10^4)$, and the space complexity is $O\left(38 + 128 + 32^2 \times 2\right) = O(2138)$.

Once the channel vector is obtained, CNN2 is applied to map it back into the channel matrix $\overline{\mathbf{H}}$. First, we input the $\mathbf{x}$ to a FC layer that reshapes it into the original size of $\overline{\mathbf{H}}$. After that, two RefineNet blocks are applied to reconstruct the channel matrix, whose architecture is shown in Fig. 3b. The first two layers of the RefineNet block are convolutional layers with the convolution kernel size of 3*3, which generate 8 and 16 feature maps, respectively. The final layer of the RefineNet block is a convolution layer, where the convolution kernel is 3*3 and the activation function is Sigmoid. As the last layer of the decoder, a convolution layer with a convolution kernel of 3*3 is applied to obtain the final refining output $\widehat{\mathbf{H}}$.

The training procedure of CNN1 and CNN2 is the same as CsiNet, which is the end-to-end learning. The loss function of the ith patch is mean square error (MSE), which is calculated as follows:

$$\mathbf{MSE} = \frac{1}{T}\sum_{j=1}^{T} ||\widehat{\mathbf{H}}_j - \overline{\mathbf{H}}_j||^2 \tag{13}$$

where T is the number of training samples in the ith patch.

3.2.2 Feature extraction and K-means++-based clustering algorithm

Clustering algorithm is a classic unsupervised machine learning algorithm. The purpose of clustering algorithm is to separate the sample points of each category and output a centroid for each category. For the codebook design, it is impossible to design the codeword for every CSI matrix. Therefore, we use the clustering algorithm to aggregate channel vectors with similar channel characteristic and design codeword for each category. We create another channel dataset for the codebook design under the same channel model, and we use CNN1 to extract the feature of dataset and obtain the channel vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_P\}$ (where P is the number of CSI matrices used for codebook design) as the input of the clustering algorithms. In other words, we cluster the channel vectors $\mathcal{X}$ into K clusters and obtain centroids $\mathcal{M}$ to design the codebook.

K-means is one of the most commonly used algorithms, and K-means++ is optimized on the basis of K-means. Different from K-means algorithm, where the initial centroids are randomly obtained, K-means++ algorithm has a more complex initialization. K-means++ algorithm can be divided into two stages: the determination of the initial centroids and the update of the centroids.

The determination of the initial centroids $\widetilde{\mathcal{M}} = \{\widetilde{\mathbf{m}}_1, \widetilde{\mathbf{m}}_2, \widetilde{\mathbf{m}}_3, \ldots, \widetilde{\mathbf{m}}_K]\}$ requires three steps:

Step 1 We choose the first initial centroid $\widetilde{\mathbf{m}}_1$ from $\mathcal{X}$ randomly.

Step 2 For each $\mathbf{x}_i (i = 1, 2, \ldots, P)$ in $\mathcal{X}$, calculate the sum distance between them from the existing initial centroids:

$$D(\mathbf{x}_i) = \sum_{j=1}^{t} d(\mathbf{x}_i, \widetilde{\mathbf{m}}_j) \tag{14}$$

where $t < K$ is the number of initial centroids we already gained, and $d(\mathbf{x}_i, \widetilde{\mathbf{m}}_j) = ||\mathbf{x}_i - \widetilde{\mathbf{m}}_j||^2$. Then, we calculate the probability that $\mathbf{x}_i$ is chosen as the next initial centroid according to:

$$P(\mathbf{x}_i) = \frac{D(\mathbf{x}_i)}{\sum_{i=1}^{P} D(\mathbf{x}_i)} \tag{15}$$

choose $\mathbf{x}_i$ with the highest probability as the next initial centroid $\widetilde{\mathbf{m}}_{t+1}$.

Step 3 Repeat Step 1 and Step 2 until we obtain K initial centroids.

Through the above three steps, we obtain the initial centroids and the distance between them is as far as possible. It can solve the problem that the random selection of the initial centroids affects K-means heavily.

Algorithm 1: K-means++ clustering algorithm

Input: K, $\mathcal{X}$
Output: $\mathcal{M}$
First stage:
Choose the first initial centroid $\widetilde{\mathbf{m}}_1$ from $\mathcal{X}$ randomly
for $i = 1$ *to* P **do**
  Calculate the sum distance from the channel vectors to the existing initial centroids:
  $D(\mathbf{x}_i) = \sum_{j=1}^{t} d(\mathbf{x}_i, \widetilde{\mathbf{m}}_j) = \sum_{j=1}^{t} ||\mathbf{x}_i - \widetilde{\mathbf{m}}_j||^2$
  Calculate the probability that $\mathbf{x}_i$ is chosen as the next initial centroid:
  $p(\mathbf{x}_i) = \frac{D(\mathbf{x}_i)}{\sum_{i=1}^{P} D(\mathbf{x}_i)}$
  $\mathbf{P} = [p(\mathbf{x}_1), p(\mathbf{x}_2), \ldots, p(\mathbf{x}_P)]$
  $i = \max_{i \in P}(\mathbf{P})$
  $\widetilde{\mathbf{m}}_{t+1} = \mathbf{x}_i$
end
Repeat 2-10 until get K initial centroids
Second stage:
$\mathcal{M} = \widetilde{\mathcal{M}}$
for $a = 1$ *to* N **do**
  Calculate the distance from the channel vectors to centroids:
  $d(a) = ||\mathbf{x}_i - \mathbf{m}_a||^2$
  $\mathbf{D} = [d(1), d(2), \ldots, d(K)]$
  $q = \min_{a \in K}(\mathbf{D})$
  Divide $\mathbf{x}_i$ into $\mathbf{Cluster}_q$ Update the centroids of each cluster:
  $\mathbf{m}_q = \underset{\mathbf{m}_q \in \mathbf{Cluster}_q}{\arg\min} \sum_{i=1}^{l} d(\mathbf{m}_q, \mathbf{x}_{q,i})$
end
Until the centroids and the channel vectors in each cluster keep unchanged
Output the final centroids $\mathcal{M} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_K\}$

After getting the initial centroids, both K-means and K-means++ use the following steps to update the initial centroids and obtain the final centroids $\mathcal{M} = \{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, ..., \mathbf{m}_K\}$. According to the nearest neighbor rule, $\mathbf{x}_i$ will be associated into qth centroid, while the distance from $\mathbf{m}_q (q = 1, 2, \ldots, K)$ to $\mathbf{x}_i$ is smallest. It can be formulated as

$$q = \underset{q=1,2,\ldots,K}{\arg\min}\, d(\mathbf{x}_i, \mathbf{m}_q) \tag{16}$$

After each $\mathbf{x}_i$ in $\mathcal{X}$ is attributable to the corresponding centroid, the channels vectors $\mathcal{X}$ can be divided into K clusters. We update the centroids of K clusters according to the following formula:

$$\mathbf{m}_q = \underset{\mathbf{m}_q \in \mathbf{Cluster}_q}{\arg\min} \sum_{i=1}^{l} d(\mathbf{m}_q, \mathbf{x}_{q,i}) \tag{17}$$

where $\mathbf{Cluster}_q$ is the qth cluster, $\mathbf{x}_{q,i}$ is the channel vector assigned to $\mathbf{Cluster}_q$, $l = 1, 2, \ldots, K$ is the number of channel vectors that $\mathbf{Cluster}_q$ contains. Repeat until the centroids and the channel vectors in each cluster keep unchanged. Finally, we get the output of clustering algorithms, the centroids $\mathcal{M}$ which are reconstructed into the codebook by CNN2.

The whole procedure is summarized in Algorithm 1.

4 Simulation results and discussion

In this section, we discuss the difference of clustering algorithms and the influence of K value selection on our proposed codebook design. Besides, we compare the proposed codebook design with the traditional DFT codebook on the sum available rates and feedback bits. We also compare the sum available rate and feedback bits between our proposed codebook-based method and the CSI feedback-based method, taking CsiNet as an example.

In OFDM system, there are $N_c = 1024$ subcarriers. We take $\widetilde{N}_c = 32$, and the size of the CNNs input layer is 32*32. There are $N_t = 32$ antennas at the BS, and the antennas adopt uniform linear array (ULA) arrangement.

Figure 4 shows the achievable rates of the proposed codebook design with different clustering algorithms: K-means and K-means++. It is noticeable that the achievable rates increase as the SNR increases from -10dB to 20 dB. As we mentioned above, the difference between the two clustering algorithms lies in the different selection method of the initial centroids. Compared with the former random selection, K-means++ adopts a more reasonable selection method to avoid falling into the local optimal solution. The simulation results also show that the K-means++ algorithm can achieve higher reachable rates when the number of centroid is the same at higher SNR. Since clustering is carried out at the BS end of the offline training stage, there will be no additional increase in computational complexity. In addition, we can see that the achievable rates increase as the number of K regardless of whether the aggregation algorithm is K-means or K-means++. It means that in our proposed algorithm, the number of codewords plays a more important role, affecting the total rate more than the choice of clustering algorithm. In other words, if we want to increase the total achievable rate, increasing the number of codewords would be our first choice in practice.

Figure 5 shows the comparison between our proposed codebook with the traditional DFT codebook. Here, we choose the K-means++ algorithm and select K=2048 codewords to improve the system performance to the highest. We also compare the sum achievable rates when the BS knows the current CSI matrix. It can be seen that with the increase in SNR, our proposed codebook has a greater advantage over the traditional DFT codebook. At the same time, the performance of our method is comparable to the

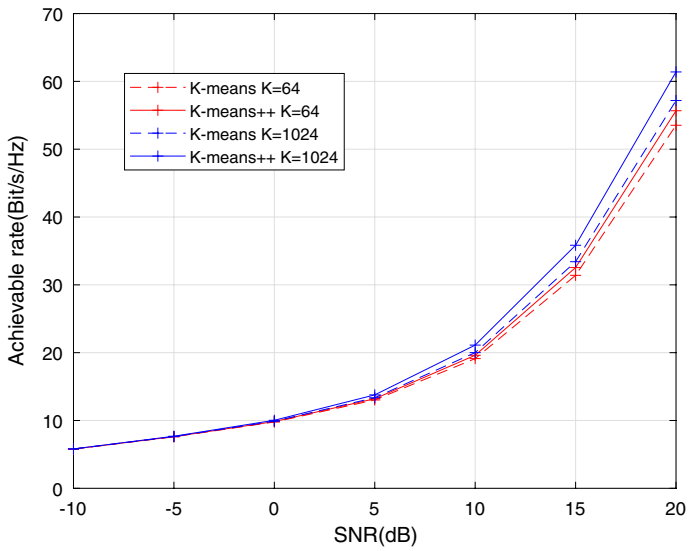

Fig. 4 Achievable rate of proposed codebook with different clustering algorithms

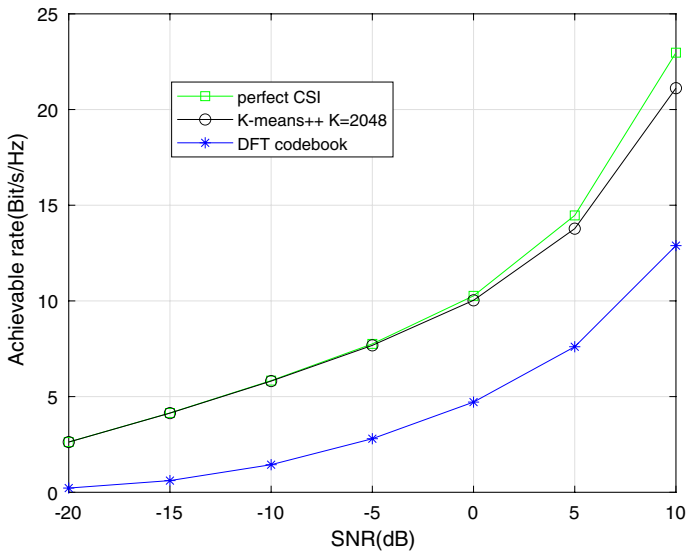

Fig. 5 Achievable rate performance of proposed codebook and DFT codebook

perfect CSI when the SNR is as low as -20dB to -5dB. In application, we can change the value of K to meet different requirements, such as fewer feedback bits or higher transmission rates.

In Fig. 6, we compared our proposed codebook with CsiNet under different r_c and the traditional DFT codebook. It is obvious that the performance of CsiNet and the proposed codebook is much better than that of the traditional DFT codebook. Meanwhile, the achievable rates of CsiNet decrease as r_c getting bigger on account of the feedback bits decrease. Our method sacrifices a small part of the available rates, which greatly reduces the number of feedback bits. Therefore, CsiNet still performs better than our proposed method even when the compression rate is pretty high.

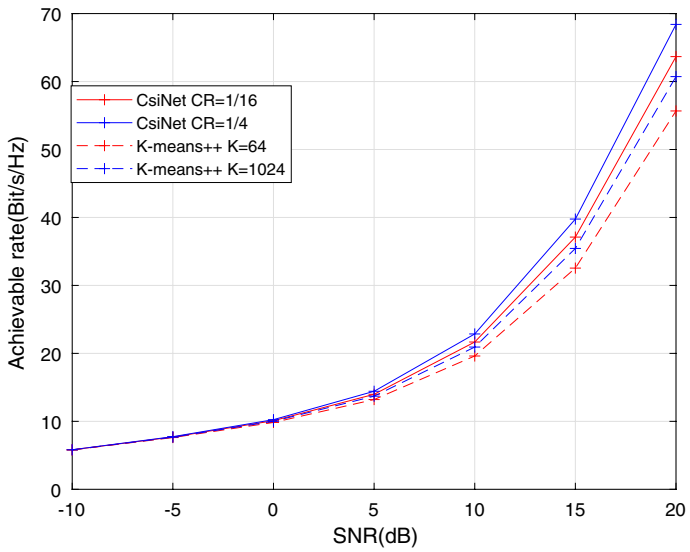

Fig. 6 Achievable rate performance of proposed codebook and CsiNet with different r_c

Table 1 Feedback bits of proposed codebook and CsiNet

K-means/K-means++		CsiNet	
K	**Feedback**	**Compact ratio**	**Feedback**
16	4	1/64	64*D
64	6	1/16	128*D
2048	11	1/4	512*D

The feedback bits of different algorithms are summarized in Table 1, where D is the quantization bits. Although CsiNet reduces the feedback bits under different compact ratio, the feedback overhead is still huge compared to the codebook-based algorithm we proposed. When $r_c = 1/16$, $D = 3$, and$K = 1024$, the feedback bits of CsiNet is 384 and the feedback bits of our proposed codebook is 10.

5 Conclusions

In this paper, we proposed a CNN and clustering-based codebook design for massive MIMO system, which significantly reduces the feedback overhead compared with the CSI feedback-based method. The proposed codebook design uses the clustering algorithm to classify the channels and obtain the centroids, whose sum distance is minimized from the real channel. The CNNs are applied to compress the CSI matrices and recover the centroids into the codewords, respectively. The simulation results show that the achievable rate of our codebook is much higher than that of traditional DFT-based codebook. While reducing the feedback bits, our proposed codebook does not sacrifice too much transmission rate compared with the CSI feedback-based method.

Abbreviations
MIMO: Massive multiple-input multiple-output; BS: Base station; CSI: Channel state information; CNN: Conventional neural network; UE: User equipment; FDD: Frequency division duplex; DFT: Discrete Fourier transform; ULA: Uniform linear arrays; DNN: Deep neural network; OFDM: Orthogonal frequency division multiplexing; SNR: Signal–noise ratio.

Acknowledgements
First, I would like to acknowledge my funders, the National Natural Science Foundation of China under Grant 61771144, which assists my research. I would particularly like to thank my supervisor, Die Hu, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

Author contributions
JX proposed the framework of the entire algorithm; simulation, analysis and interpretation of the results. DH participated in the conception of this research and revised the manuscript. All authors have read and approved the final manuscript.

Funding
This research is supported by the National Natural Science Foundation of China under Grant 61771144.

Availability of data and materials
Please contact author for data requests.

Declarations

Ethical approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 21 November 2021 Accepted: 22 May 2022
Published online: 04 June 2022

References

1. E.G. Larsson, O. Edfors, F. Tufvesson, T.L. Marzetta, Massive mimo for next generation wireless systems. IEEE Commun. Mag. **52**(2), 186–195 (2014)
2. F. Rusek, D. Persson, B.K. Lau, E.G. Larsson, T.L. Marzetta, O. Edfors, F. Tufvesson, Scaling up MIMO: opportunities and challenges with very large arrays. IEEE Signal Process. Mag. **30**(1), 40–60 (2012)
3. Z. Jiang, A.F. Molisch, G. Caire, Z. Niu, Achievable rates of FDD massive MIMO systems with spatial channel correlation. IEEE Trans. Wirel. Commun. **14**(5), 2868–2882 (2015)
4. H. Sun, Z. Zhao, X. Fu, M. Hong, Limited feedback double directional massive MIMO channel estimation: From low-rank modeling to deep learning, in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5 (2018)
5. J. Guo, C.-K. Wen, S. Jin, G.Y. Li, Convolutional neural network-based multiple-rate compressive sensing for massive MIMO CSI feedback: design, simulation, and analysis. IEEE Trans. Wirel. Commun. **19**(4), 2827–2840 (2020)
6. M.B. Mashhadi, Q. Yang, D. Gündüz, CNN-based analog CSI feedback in FDD MIMO-OFDM systems, in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8579–8583 (2020)
7. H. Noh, Y. Kim, J. Lee, C. Lee, Codebook design of generalized space shift keying for FDD massive MIMO systems in spatially correlated channels. IEEE Trans. Veh. Technol. **64**(2), 513–523 (2014)
8. D.J. Love, R.W. Heath, Equal gain transmission in multiple-input multiple-output wireless systems. IEEE Trans. Commun. **51**(7), 1102–1110 (2003)
9. D. Yang, L.-L. Yang, L. Hanzo, DFT-based beamforming weight-vector codebook design for spatially correlated channels in the unitary precoding aided multiuser downlink, in *2010 IEEE International Conference on Communications*, pp. 1–5 (2010)
10. J. Jiang, X. Wang, G.A.S. Sidhu, L. Zhen, R. Gao, Clustering-based codebook design for MIMO communication system, in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6 (2019)
11. E. Balevi, A. Doshi, J.G. Andrews, Massive MIMO channel estimation with an untrained deep neural network. IEEE Trans. Wirel. Commun. **19**(3), 2079–2090 (2020)
12. M.S. Sim, J. Park, C.-B. Chae, R.W. Heath, Compressed channel feedback for correlated massive MIMO systems. J. Commun. Netw. **18**(1), 95–104 (2016)
13. N. Ravindran, N. Jindal,H.C. Huang, Beamforming with finite rate feedback for LOS MIMO downlink channels, in *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pp. 4200–4204 (2007)
14. J. Jiang, X. Wang, W.-J. Wang, L. Zhen, J. Wang, Deep clustering-based codebook design for massive MIMO systems. IEEE Access **7**, 172654–172664 (2019)
15. L. Liu, C. Oestges, J. Poutanen, K. Haneda, P. Vainikainen, F. Quitin, F. Tufvesson, P. De Doncker, The cost 2100 MIMO channel model. IEEE Wirel. Commun. **19**(6), 92–99 (2012)

16. P. Liang, J. Fan, Deep learning and compressive sensing-based CSI feedback in FDD massive MIMO systems. IEEE Trans. Veh. Technol. **69**(8), 9217–9222 (2020)
17. C.-K. Wen, W.-T. Shih, Deep learning for massive MIMO CSI feedback. IEEE Wirel. Commun. Lett. **7**(5), 748–751 (2018)

Publisher's Note