LETTER

CsiNet-Plus Model with Truncation and Noise on CSI Feedback

Feng LIU[†], **Xuecheng HE**[†], **Conggai LI**[†a)], ***Nonmembers*, *and* Yanli XU**[†], ***Member***

SUMMARY For the frequency-division-duplex (FDD)-based massive multiple-input multiple-output (MIMO) systems, channel state information (CSI) feedback plays a critical role. Although deep learning has been used to compress the CSI feedback, some issues like truncation and noise still need further investigation. Facing these practical concerns, we propose an improved model (called CsiNet-Plus), which includes a truncation process and a channel noise process. Simulation results demonstrate that the CsiNet-Plus outperforms the existing CsiNet. The performance interchangeability between truncated decimal digits and the signal-to-noise-ratio helps support flexible configuration.

key words: *massive MIMO, CSI feedback, truncation, channel noise, deep learning*

1. Introduction

With the continuous development of communication technologies, the demand for high-speed networks is growing stronger. As a key technology of 5G, massive multiple-input multiple-output (MIMO) technology is receiving more and more attentions in the industry, and it is considered as the key to improving network speed. For massive MIMO systems, channel state information (CSI) plays a crucial role of the system, which can increase channel capacity, improve bit error rate, and reduce hardware complexity. In the frequency-division-duplex (FDD) mode, the user equipment (UE) needs to send its CSI back to the base station (BS), and the BS adjusts the system transmission policy according to the CSI. In order to guarantee timely CSI, the feedback delay must be less than the coherence time. Thus, MIMO systems need to deliver CSI on time to ensure real-time performance, especially in rapidly changing channels. However, as the number of antennas increases, the data volume of feedback CSI also increases, which leads to high demands placed on the system. Therefore, it is crucial to propose a feasible compression algorithm.

Compression sensing (CS) has been proposed to solve the compression problem of large-scale antenna feedback channels [1] which caused widespread concerns. However, this model is not suitable for practical channel transmission due to its high complexity. In 2018, [2] proposed the CsiNet neural network model for compressed CSI feedback of massive MIMO with spatial redundancy, while [3] further proposed the LSTM-NET to compress the CSI with temporal redundancy. In 2019, [4] proposes a combination of LSTM module and FCN in the neural network architecture for the problem of insufficient tracking time correlation [2], [3]. Although these models have excellent feedback efficiency, many scholars have further improved it from different aspects. [5] proposes DualNet-MAG and DualNet-ABS to improve spectral efficiency using bi-directional channel reciprocity. [6] uses the idea of combining deep learning and superposition coding to superimpose CSI on the BS's uplink user data sequence. [7] proposes a CSI compression feedback algorithm based on deep learning, which is suitable for single-user and multi-user scenarios. [8] proposes a new deep AutoEncoder which considers the feedback error and feedback delay. And DeepCMC scheme [9] for processing CSI of different dimensions is used to further compress the encoded data using quantization and entropy coding.

Nevertheless, the issues of truncation and channel noise during the CSI feedback should be further investigated. For the truncation issue, because the encoded data is often float, the average code length (ACL) will increase rapidly along with the number of float data digits. Although [9] uses uniform quantization to reduce feedback overhead, the impact of quantization precision is not considered. The other issue is about the channel noise, which is inevitable in practical communication but has not been considered for the CS-based CSI feedback. The influence of channel noise on the system performance should also be measured. Overall, how to truncate the data in variant decimal digits and adapt to different channel noise level is an urgent and important problem to be addressed.

In this letter, we propose a CsiNet-Plus model based on the existing CsiNet to solve the above problem. In response to the truncation issue, in 2017, Twitter researcher Lucas proposed a way to solve its back propagation in image compression [10]. For the channel noise issue, Sebastian proposed a two-step optimization method to solve the problem of channel joint processing [11]. Motivated by the ideas of these two articles, we can solve the above CSI feedback problem with truncation and noise. Entropy coding is used to reduce the feedback overhead with truncated data in different numbers of digits. A two-step optimization algorithm is used to train the data with channel noise. The impacts on the ACL and system performance are compared by simulation. The performance interchangeability between the truncated decimal digits and the signal-to-noise-ratio (SNR) is demonstrated, which can be used for flexible configuration.

Manuscript received August 30, 2019.

[†]The authors are with the College of Information Engineering, Shanghai Maritime University, Shanghai, China.

a) E-mail: cgli@shmtu.edu.cn

DOI: 10.1587/transfun.2019EAL2123

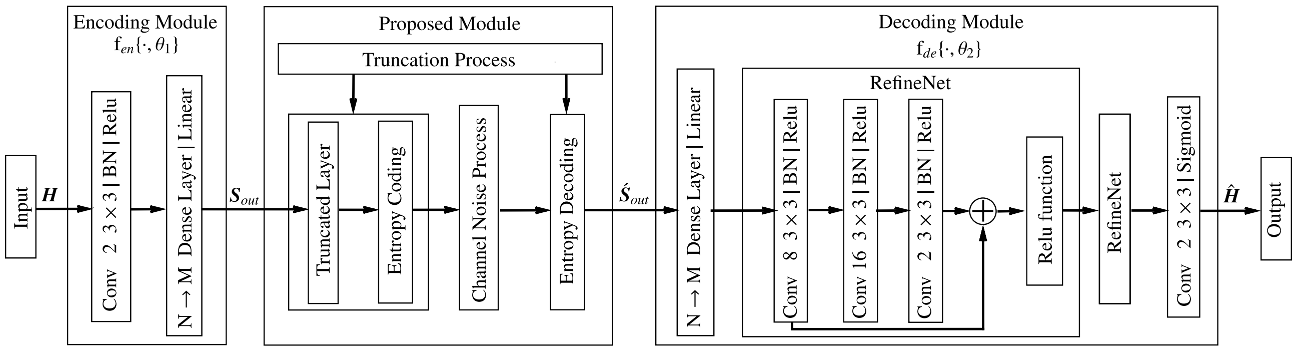

Fig. 1 CsiNet-Plus model, including the encoding/decoding, and proposed modules.

2. System Model

We assume a simple single-cell single-user downlink massive MIMO system with FDD mode. There are N_t antennas at BS and a single antenna at the UE. The system is operated in orthogonal frequency division multiplexing (OFDM) over $\tilde{N}_c$ subcarriers. The CSI data $\tilde{\boldsymbol{H}}$ in frequency-spatial domain has a dimension of $\tilde{N}_c \times N_t$, which should be transmitted back to the BS side. To reduce the feedback amount of CSI, we follow the framework of [2] using the deep learning method.

Figure 1 shows the system model of CsiNet-Plus scheme, where the middle box indicates the proposed module. The first and third boxes respectively represent the encoding and decoding modules given by [2]. The data compression is divided into two steps.

Firstly, the original channel coefficient matrix $\tilde{\boldsymbol{H}}$ can be converted by 2D inverse discrete Fourier transform (IDFT)[†] for initial compression, as shown below

$$\boldsymbol{H}_0 = \frac{1}{\tilde{N}_c \times N_t} \boldsymbol{F}_d \tilde{\boldsymbol{H}} \boldsymbol{F}_a^H \tag{1}$$

where $\boldsymbol{F}_d$ and $\boldsymbol{F}_a$ are $\tilde{N}_c \times \tilde{N}_c$ and $N_t \times N_t$ IDFT matrices, respectively. And $(\cdot)^H$ is the Hermitian operation on a vector/matrix. Due to limited multipath time delay, performing 2D-IDFT can transform $\tilde{\boldsymbol{H}}$ into a sparsed matrix, with only the first $N_c (< \tilde{N}_c)$ rows having distinct non-zero values. This operation will give a $\tilde{N}_c \times N_t$ matrix. For simplicity, we directly use $\boldsymbol{H}$ to denotes the first N_c rows of $\boldsymbol{H}_0$.

With $\boldsymbol{H}$ as input, the system uses an improved version of CsiNet (referred to as CsiNet-Plus) to further compress the data. The encoding module is mainly composed of one convolution layer and one dense layer with linear activation function, among which the former is used to extract features of the original data and the latter is used to extract compressed data. The convolutional layer uses a 3×3 kernel to generate two feature maps. Then we uses batch normalization (BN) and relu functions (Relu) to prevent overfitting.

[†]DFT is performed here in [2]. However, we find that the inverse operation of DFT and IDFT will obtain a better performance. So we use this inversed version in this letter.

The encoding module generates the following output

$$\boldsymbol{S}_{out} = \mathrm{f}_{\mathrm{en}}\{\boldsymbol{H}, \theta_1\} \tag{2}$$

where θ_1 is the encoding parameter set to be determined by training and $\mathrm{f}_{\mathrm{en}}\{\cdot\}$ is the encoding module. Define the compression ratio as $\lambda = M/N$, which M and N represents the row dimension of $\boldsymbol{S}_{out}$ and $\boldsymbol{H}$, respectively. So $\boldsymbol{S}_{out}$ is a $\lambda N_c \times N_t$ matrix.

Then the encoded data is handled by the proposed module, which is composed by truncation and channel noise processes, as shown in Fig. 1. For the former process, there are three steps including truncation layer, entropy coding and decoding, while the latter adds noise into the channel before entropy decoding. This module produces the output $\acute{\boldsymbol{S}}_{out}$ with the same dimension as $\boldsymbol{S}_{out}$.

After that, the decoding module will provide a reverse function of the encoding module. It consists mainly of one dense layer, one convolution layer and two RefineNet [2], where RefineNet is based on residual neural networks [12]. The dense layer is used to expand M-dimensional data to N-dimension, while RefineNet is used to restore data. The output gives an estimation of $\boldsymbol{H}$ with a dimension of $N_c \times N_t$

$$\hat{\boldsymbol{H}} = \mathrm{f}_{\mathrm{de}}\{\acute{\boldsymbol{S}}_{out}, \theta_2\} \tag{3}$$

where θ_2 is the decoding parameter set to be determined by training and $\mathrm{f}_{\mathrm{de}}\{\cdot\}$ is the decoding module.

Finally, the estimation of the original channel data $\tilde{\boldsymbol{H}}$ can be obtained by performing 2D-DFT as

$$\hat{\tilde{\boldsymbol{H}}} = \boldsymbol{F}_e \hat{\boldsymbol{H}}_0 \boldsymbol{F}_f^H \tag{4}$$

where $\hat{\boldsymbol{H}}_0$ is the $\tilde{N}_c \times N_t$ matrix after dimension expansion with zero padding from $\hat{\boldsymbol{H}}$, while $\boldsymbol{F}_e$ and $\boldsymbol{F}_f$ are $\tilde{N}_c \times \tilde{N}_c$ and $N_t \times N_t$ DFT matrices, respectively.

The powerful ability of deep learning can improve the compression performance of the system, but it does not consider the communication problems that occur in the actual transmission process, such as the truncation of decimal digits and channel noise in the feedback transmission.

1. In a normal deep learning system, the data storage type is float. The original information passes through the encoding module and the format of the output is float, too.

So the decimal digits will play an important role when considering the compress problem to reduce the feedback amount. Obviously, if the more decimal digits of the data are truncated, the less ACL will be obtained at the cost of accuracy degradation.

2. In the actual transmission process, encoding data needs to be transmitted through the channel with additive noise. Different SNR has different effects on the performance of the feedback system. However, existing models like [2] and [3] ignore the impact of channel noise on the system, i.e., $\boldsymbol{S}_{out} = \acute{\boldsymbol{S}}_{out}$, which is an ideal scenario. With channel noise, the recovery accuracy of the decoding module will be highly dependent on the accuracy of $\acute{\boldsymbol{S}}_{out}$.

In this paper, we solve the above two issues by adding the proposed module between the encoding and decoding modules. The detail will be given in the next section.

3. Proposed CsiNet-Plus Scheme

To measure the effect of truncation and/or channel noise, we will consider the following three scenarios.

3.1 Only Truncation without Channel Noise

Without channel noise, the entropy encoding and decoding modules will be directly connected and only truncation operation is left. The truncated layer performs decimal digit truncation of $\boldsymbol{S}_{out}$ denoted by

$$\acute{\boldsymbol{S}}_{out} = \boldsymbol{S}_{tr}^{d} = \mathrm{T}\{\boldsymbol{S}_{out}, d\} \tag{5}$$

where $\mathrm{T}\{\cdot, d\}$ is truncation operation and d (called truncation length) is number of decimal digits to be kept by rounding operation on the $d+1$ th decimal digit. By controlling d, the ACL in entropy coding can be adjusted. A smaller d can reduce the ACL and improve the feedback efficiency.

In a neural network, the system calculates the output by propagating the data in the forward direction. This output is then compared to the expected result to yield a mean square error (MSE)

$$Err = MSE(\hat{\boldsymbol{H}}, \boldsymbol{H}) = \frac{1}{N_c N_t} \|\hat{\boldsymbol{H}} - \boldsymbol{H}\|_F^2 \tag{6}$$

where $\|\boldsymbol{A}\|_F$ indicates the Frobenius norm of matrix $\boldsymbol{A}$. Through back-propagation [13], this error is reversed back to the original system to reduce the error value by adjusting the parameters of each layer.

In the traditional communications, truncating data is a very common operation. But in neural networks, this can cause significant problem because we cannot calculate the gradient of the truncation process. So the gradient cannot be directly back-propagated. Here we will use the scheme proposed by [10] to solve this problem. By regarding truncation error as additive noise, we have the following approximation

$$\boldsymbol{S}_{tr} \approx \boldsymbol{S}_{out} + \boldsymbol{u} \tag{7}$$

The gradient of the truncation process can be computed as

$$\frac{d}{ds_{out}} \boldsymbol{S}_{tr} \approx \frac{d}{ds_{out}} (\boldsymbol{S}_{out} + \boldsymbol{u}) = 1 \tag{8}$$

After truncation layer, the probability distribution of $\boldsymbol{S}_{tr}$ will be changed in comparison with that of $\boldsymbol{S}_{out}$. Thus we further use the entropy encoding like Huffman algorithm to reduce the ACL without any performance loss. Accordingly, entropy decoding will be implemented at the base station.

3.1.1 Parameter Training

To obtain the encoding/decoding parameters $\hat{\theta}_1^d$ and $\hat{\theta}_2^d$, we give the following training process for the above scenario to minimize the MSE estimation of $\boldsymbol{H}$

$$(\hat{\theta}_1^d, \hat{\theta}_2^d) = \underset{\theta_1, \theta_2}{\arg\min} \|\mathrm{f}_{\mathrm{de}}\{\mathrm{T}\{\mathrm{f}_{\mathrm{en}}\{\boldsymbol{H}, \theta_1\}, d\}, \theta_2\} - \boldsymbol{H}\|_F^2 \tag{9}$$

3.2 Only Channel Noise without Truncation

Without truncation, there is no truncation related process such as the entropy encoding/decoding. The noise is assumed to be additive Gaussian distributed and not truncated. The input-output relationship is

$$\acute{\boldsymbol{S}}_{out} = \boldsymbol{S}_{ns}^{k} = \mathrm{N}\{\boldsymbol{S}_{out}, \boldsymbol{n}^k\} = \boldsymbol{S}_{out} + \boldsymbol{n}^k \tag{10}$$

where $\boldsymbol{n}^k$ is the additive noise with $SNR = k$, $\mathrm{N}\{\cdot, \boldsymbol{n}^k\}$ is used to denote the channel noise process.

3.2.1 Parameter Training

To determine the training parameter, if we take the channel noise process as a layer that does not affect the system and set its back propagation gradient to a constant, the optimization speed of the model will be greatly affected, since there are too many external operations like modulation and demodulation which cannot be speed up by the GPU resource. Thus we turn to the two-step optimization [11] and [14] for training the encoding and decoding parameters.

In the first step, $\hat{\theta}_1$ and $\hat{\theta}_2$ are trained by minimizing the MSE of $\boldsymbol{H}$ in the ideal setting without channel noise as

$$(\hat{\theta}_1, \hat{\theta}_2) = \underset{\theta_1, \theta_2}{\arg\min} \|\mathrm{f}_{\mathrm{de}}\{\mathrm{f}_{\mathrm{en}}\{\boldsymbol{H}, \theta_1\}, \theta_2\} - \boldsymbol{H}\|_F^2 \tag{11}$$

In the second step, $\hat{\theta}_1$ is fixed and $\hat{\theta}_2^k$ is further trained according to $\hat{\theta}_2$ and $\boldsymbol{n}^k$

$$\hat{\theta}_2^k = \underset{\hat{\theta}_2}{\arg\min} \|\mathrm{f}_{\mathrm{de}}\{\mathrm{N}\{\boldsymbol{S}_{out}, \boldsymbol{n}^k\}, \hat{\theta}_2\} - \boldsymbol{H}\|_F^2 \tag{12}$$

The specific steps are shown by Algorithm 1.

By training the decoding parameter set $\hat{\theta}_2^k$ under different k, we can obtain the candidates of parameter settings with variant SNR level.

Algorithm 1 Two-step training for the scenario with only channel noise

```
Input: Training Sets: $D = (\boldsymbol{H}, \boldsymbol{H})$; epochs $T = 1000$; $SNR = k$
Output: $\hat{\theta}_1, \hat{\theta}_2^k$
 1: Step 1. Determine the encoding parameters $\hat{\theta}_1$
 2: Initialize. Randomly choose $\theta_1$ and $\theta_2$
 3: for $t = 1 \to T$ do
 4:     Update $\hat{\theta}_1$ and $\hat{\theta}_2$ by training the CsiNet to compute Eq.(11)
 5: end for
 6: Get $S_{out}$ by computing $f_{en}\{H, \hat{\theta}_1\}$.
 7: Step 2. Determine the decoding parameters $\hat{\theta}_2$
 8: Initialize. Use $\hat{\theta}_2$ obtained in Step 1 for initialization
 9: Get $\boldsymbol{S}_{ns}^k$ by computing Eq.(10).
10: for $t = 1 \to T$ do
11:     Update $\hat{\theta}_2^k$ by training decoding module to compute Eq.(12)
12: end for
13: return $\hat{\theta}_1, \hat{\theta}_2^k$
```

Table 1 Performance with different truncation digits.

DIGIT	1	2	3	4	5	6	7	8
ACL	3.7	6.6	7.6	8.0	8.3	8.4	8.5	8.6
NMSE	−17.17	−20.04	−20.15	−20.19	−20.43	−20.69	−20.71	−21.56
ρ	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 2 Performance comparison with different SNR.

SNR	2	4	6	8	10	12	14
Existing CsiNet							
NMSE	13.99	12.03	8.48	2.79	-5.89	-16.60	-20.69
ρ	0.21	0.33	0.58	0.85	0.97	0.99	0.99
Proposed CsiNet-Plus							
NMSE	−2.18	−5.27	−8.87	−12.99	−16.50	−20.76	−21.34
ρ	0.66	0.85	0.93	0.97	0.98	0.99	0.99

Table 3 Performance with truncation and channel noise.

The effects of truncation and channel noise on *NMSE*							
SNR / Digit	2	4	6	8	10	12	14
2	0.23	−2.01	−5.04	−9.41	−12.45	−19.01	−19.02
4	−1.11	−4.04	−7.11	−10.89	−14.09	−19.06	−20.19
6	−1.51	−5.14	−8.66	−11.79	−15.68	−19.74	−20.61
8	−2.18	−5.17	−8.82	−12.55	−16.30	−20.05	−20.91
The effects of truncation and channel noise on ρ							
SNR / Digit	2	4	6	8	10	12	14
2	0.34	0.64	0.83	0.93	0.97	0.99	0.99
4	0.54	0.78	0.89	0.95	0.97	0.99	0.99
6	0.61	0.84	0.93	0.96	0.98	0.99	0.99
8	0.67	0.85	0.93	0.96	0.98	0.99	0.99

3.3 Simultaneous Truncation and Channel Noise

As shown in Fig. 1, the proposed module simultaneously considers the truncation and the channel noise processes.

Denote the truncation length by d_0 and SNR by k_0, the input-output relationship of the proposed module is

$$\acute{\boldsymbol{S}}_{out} = \mathrm{N}\{\boldsymbol{S}_{tr}^{d_0}, \boldsymbol{n}^{k_0}\} = \mathrm{N}\{\mathrm{T}\{\boldsymbol{S}_{out}, d_0\}, \boldsymbol{n}^{k_0}\} \tag{13}$$

Then the CSI estimation is given by

$$\hat{\boldsymbol{H}} = \mathrm{f_{de}}\{\mathrm{N}\{\mathrm{T}\{\mathrm{f_{en}}\{\boldsymbol{H}, \theta_1\}, d_0\}, \boldsymbol{n}^{k_0}\}, \theta_2\} \tag{14}$$

3.3.1 Parameter Training

As mentioned earlier, here we still use the two-step method to speed up the parameter training.

Firstly, $\hat{\theta}_1^{d_0}$ and $\hat{\theta}_2^{d_0}$ are determined by

$$(\hat{\theta}_1^{d_0}, \hat{\theta}_2^{d_0}) = \underset{\theta_1, \theta_2}{\arg\min} \|\mathrm{f_{de}}\{\mathrm{T}\{\mathrm{f_{en}}\{\boldsymbol{H}, \theta_1\}, d_0\}, \theta_2\} - \boldsymbol{H}\|_F^2 \tag{15}$$

And we get truncated encoded data $\boldsymbol{S}_{tr}^{d_0}$ by Eq. (5).

Secondly, we add the channel noise process $\mathrm{N}\{\boldsymbol{S}_{tr}^{d_0}, \boldsymbol{n}^{k_0}\}$ and get decoding parameters $\hat{\theta}_2^{d_0,k_0}$, as shown below

$$\hat{\theta}_2^{d_0,k_0} = \underset{\hat{\theta}_2^{d_0}}{\arg\min} \|\mathrm{f_{de}}\{\mathrm{N}\{\boldsymbol{S}_{tr}^{d_0}, \boldsymbol{n}^{k_0}\}, \hat{\theta}_2^{d_0}\} - \boldsymbol{H}\|_F^2 \tag{16}$$

The detail is similar with Algorithm 1 but omitted here due to space limitation. With this training method, the system parameters $\hat{\theta}_1^{d_0}$ and $\hat{\theta}_2^{d_0,k_0}$ can be determined.

4. Simulation Results

We use the same indoor data as [2] for experiments, which is obtained by simulation through COST2100 [15]. The BS has $N_t = 32$ antennas and $\tilde{N}_c = 1024$ sub-carriers are set for OFDM. After the system uses the 2D-IDFT transform of the original data $\tilde{\boldsymbol{H}}$, the first $N_c = 32$ rows of data $\boldsymbol{H}$ are retained. The compression ratio is set to be $\lambda = 1/4$.

Performance is measured by $NMSE$ and ρ. The difference between the recovered channel $\hat{\boldsymbol{H}}$ and original $\boldsymbol{H}$ is quantified by $NMSE$, which is defined as:

$$NMSE \triangleq E\{\|\boldsymbol{H} - \hat{\boldsymbol{H}}\|_F^2 / \|\boldsymbol{H}\|_F^2\} \tag{17}$$

The cosine similarity between the original channel vector $\tilde{\boldsymbol{h}}_n$ at the nth subcarrier over all antennas and its estimation $\hat{\tilde{\boldsymbol{h}}}_n$ from 2D-DFT of $\hat{\boldsymbol{H}}$ is described by

$$\rho \triangleq E\left\{\frac{1}{\tilde{N}_c} \sum_{n=1}^{\tilde{N}_c} \frac{|\hat{\tilde{\boldsymbol{h}}}_n^H \tilde{\boldsymbol{h}}_n|}{\|\hat{\tilde{\boldsymbol{h}}}_n\|_2 \|\tilde{\boldsymbol{h}}_n\|_2}\right\} \tag{18}$$

We conducted three sets of experiments to study the effects of truncation, and/or channel noise. The training, testing and validation sets consist of 100,000, 20,000 and 30,000 samples, respectively. During the training, we use the Adam optimizer with MSE to calculate the encoding and decoding parameters. The epochs, learning rate and batch size are set as 1000, 0.001 and 200, respectively.

According to the three scenarios in the precious section, simulation results are correspondingly shown below.

4.1 The Effect of Truncation without Channel Noise

From the simulation we find that about 80% of the original data $\boldsymbol{S}_{out}$ uses 8 decimal digits, while the other 20% has a longer digits. The proposed scheme sets the truncation length from 1 to 8. The simulation results are provided in Table 1. It can be seen from Table 1 that the number

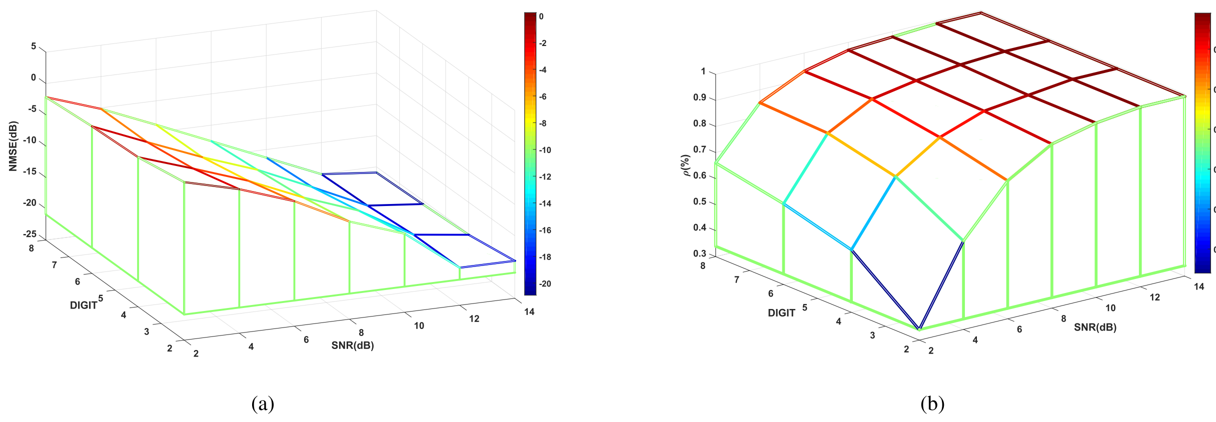

Fig. 2 The effects of truncation and channel noise on system performance with (a) $NMSE$; (b) ρ.

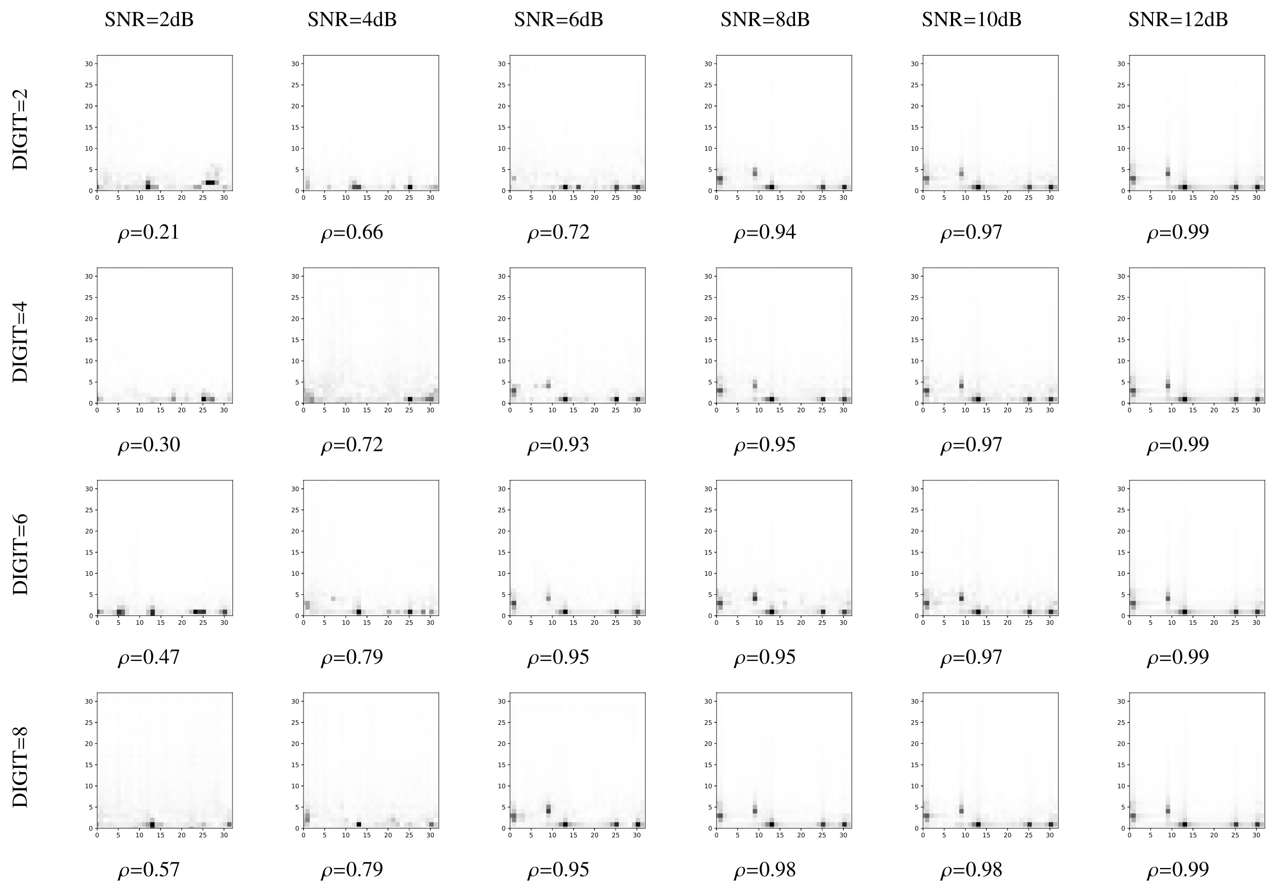

Fig. 3 Reconstructed images and ρ with different truncated decimal digits and SNRs.

of truncated digits has a significant influence on the ACL. However, truncated digits has a much less impact on performance metrics. As the number of digits increases, the ACL also becomes larger along with a slightly better performance such as $NMSE$ and ρ. Specially, we find that when the number of truncated bits is greater than 1, the truncation error has little effect on the system recovery.

4.2 The Effect of Channel Noise without Truncation

Performance comparison with different SNR in a range of 2–14 dB is shown in Table 2. We can see that the proposed CsiNet-Plus outperforms the existing CsiNet from both $NMSE$ and ρ aspects, especially in the low SNR range.

4.3 The Effect of Truncation and Channel Noise

The effects of simultaneous truncation and channel noise on $NMSE$ and ρ are shown in Table 3 and Fig. 2. From both we can observe that as the SNR or the truncated digits become larger, the recovery quality gradually becomes better. So it exhibits an interchangeability between the SNR and the truncated decimal digits. For example, with a higher SNR, the number of truncated digits can be smaller for the same performance. When the SNR is large enough, the truncation number can be very small. However, low SNR often causes serious performance degradation even with a large truncation number. In contrast with the existing CsiNet model with performance shown in Table 2, the proposed scheme shows improvement even with a very short truncation length of $d_0 = 2$. Therefore, the proposed CsiNet-Plus really works and performs better than the existing CsiNet. We remark that the data in Table 3 with a truncation length of 8 digits is slightly worse than that in Table 2 since there is no truncation process in the later scenario.

Figure 3 shows some reconstruction samples at different SNR levels and truncated decimal digits along with the corresponding pseudo-gray plots of the strength of $\hat{\boldsymbol{H}}$. We can use the image with $SNR = 12$ and $DIGIT = 8$ as the reference image. By observing the difference between these figures and the reference image, we find that when the difference is small, the ρ is large. Again, with a higher SNR or more truncated digits, better performance can be obtained. This snapshot demonstrates coincident with the statistical behavior as Table 3 and Fig. 2.

5. Conclusion

For practical applications, we proposed the CsiNet-Plus model to adapt the situation of truncation and channel noise in the CSI feedback transmission. The performance in different truncation digits and channel noise level are evaluated, from which an interchangeable relationship between them can be observed. The proposed scheme is shown to outperforms the existing CsiNet.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 61271283, U1701265, 61472237, 61601283).

References

[1] P. Kuo, H.T. Kung, and P. Ting, “Compressive sensing based channel feedback protocols for spatially-correlated massive antenna arrays,” 2012 IEEE Wireless Communications and Networking Conference (WCNC), pp.492–497, April 2012.

[2] C. Wen, W. Shih, and S. Jin, “Deep learning for massive MIMO CSI feedback,” IEEE Wireless Commun. Lett., vol.7, no.5, pp.748–751, Oct. 2018.

[3] T. Wang, C. Wen, S. Jin, and G.Y. Li, “Deep learning-based CSI feedback approach for time-varying massive MIMO channels,” IEEE Wireless Commun. Lett., vol.8, no.2, pp.416–419, Oct. 2018.

[4] 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.

[5] 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.

[6] C. Qing, B. Cai, Q. Yang, J. Wang, and C. Huang, “Deep learning for CSI feedback based on superimposed coding,” IEEE Access, vol.7, pp.93723–93733, 2019.

[7] Y. Liao, H. Yao, Y. Hua, and C. Li, “CSI feedback based on deep learning for massive MIMO systems,” IEEE Access, vol.7, pp.86810–86820, 2019.

[8] 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.

[9] Q. Yang, M.B. Mashhadi, and D. Gündüz, “Deep convolutional compression for massive MIMO CSI feedback,” arXiv preprint arXiv:1907.02942, 2019.

[10] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” arXiv preprint arXiv:1703.00395, 2017.

[11] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, “Deep learning based communication over the air,” IEEE J. Sel. Topics Signal Process., vol.12, no.1, pp.132–143, 2018.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proc. IEEE conference on computer vision and pattern recognition, pp.770–778, 2016.

[13] D. Rumerlhar, “Learning representation by back-propagating errors,” Nature, vol.323, pp.533–536, 1986.

[14] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, “An end-to-end compression framework based on convolutional neural networks,” IEEE Trans. Circuits Syst. Video Technol., vol.28, no.10, pp.3007–3018, Oct. 2018.

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