Performance Evaluation of Channel Decoder based on Recurrent Neural Network

Saisi Meng[1,a], Xue-Qin Jiang[1,b] , Yongbin Gao[2,c], Han Hai[1,d] and Jia Hou[3,e]

[1]School of Information Science and Technology, Donghua University, Shanghai, China
[2]School of Engineering Science, Shanghai University, Shanghai, China
[3]School of Electronics and Information, Soochow University, Suzhou, China

Email: [a]saisi_meng@163.com, [b]xqjiang@dhu.edu.cn, [c]gaoyongbin@sues.ehu.cn, [d]hhai@dhu.edu.cn, [e]houjia@suda.edu.cn

Abstract. Recently, recurrent neural network (RNN) has demonstrated superior performance for channel decoder, which motivates us to explore which RNN decoder can be more efficient. In this paper, we propose three kinds of RNN decoders, which are built upon long short term memory (LSTM), gated recurrent unit (GRU) and bidirectional gated recurrent units (Bi-GRU), respectively. The performance of these three RNN decoders are evaluated through lots of simulations, which indicate that the GRU decoder with simplest structure and least computational time, has similar bit error rate (BER) performance as that of the LSTM decoder. The Bi-GRU decoder has the best BER performance at the expenses of more computational time. However, it is prone to overfitting. Furthermore, we find that the BER performance of RNN decoders without dropout is better than that of the decoders with dropout when decoding models are underfitting, while the RNN decoders are better to dropout when decoding models are overfitting.

1. Introduction

With the improvement of fifth generation (5G) network technology, a new generation of communication mode is coming soon. The theoretical transmission speed of 5G networks, which is hundreds of times faster than 4G, can reach tens of Gbs per second. Obviously, higher transmission rate requires lower decoding latency. Since most of the decoding algorithms involves large numbers of iterative calculations, we need to design a decoder that should satisfy the demand of high-speed and low-latency. It happens that deep learning can help us solve this problem.

Deep learning has shown great potential and advantage in some complex tasks. It has achieved remarkable results in image processing [1], machine translation [2], speech recognition [3] and many other areas. Inspired by powerful learning ability of deep learning, [4] provided some evidence that structured codes were easier to learn than random codes, and pointed out that the neural networks could learn a structure of decoding algorithm, rather than a simple classifier. [5] compared three types of neural network decoder (NND) with the same parameter magnitude and drew a conclusion that recurrent neural network (RNN) decoder had the best decoding BER performance but the highest computational time. In recent years, RNN has derived many variants, which demonstrate better performance in many tasks. According to the conclusion of [5], we want to explore the performance of

the decoders built upon RNN variants. In the following works, when the decoders of these RNN variants appear at the same time, we collectively refer to them as RNN decoders.

In this paper, we propose three kinds of RNN decoders, which are built upon long short-term memory (LSTM), gated recurrent unit (GRU) and bidirectional gated recurrent unit (Bi-GRU). We discuss them separately in the case of decoders with dropout and decoders without dropout. In this work we focus on decoding polar codes, which are mathematically proven to achieve channel capacity. Our goal for this paper is to find which RNN decoder can be more efficient, which is measured by two indicators, decoding BER and computational time.

2. System design

2.1. Channel model

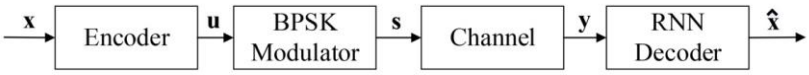

Figure 1. The architecture of channel model with RNN decoder.

The architecture of channel model with RNN decoder is illustrated in figure 1. The task of the encoder at the transmitter is to encode information bits $\mathbf{x}$ of length K into a binary codeword $\mathbf{u}$ of length N. We assume that through binary phase shift keying (BPSK) modulation, the codeword $\mathbf{u}$ can be mapped to a symbol vector **s**. After transmitting the symbol vector **s** over a channel with the additive white Gaussian noise (AWGN), a noisy version of the codeword **y** is received at the receiver. The received vector **y** can be written as $\mathbf{y} = \mathbf{s} + \mathbf{n}$, where $\mathbf{n}\sim\mathrm{N}(\mathbf{0}, \sigma^2 I_N)$ represents the N × 1 symbol vector. The task of the decoder is to recover the vector **y** to the corresponding information bits **x**. Here, we denote the recovered bits $\hat{\mathbf{x}}$. We hope that after decoding by the RNN decoders, the BER can be as close as possible to the maximum a posteriori (MAP) decoding. For details we refer to [5].

2.2. Training

The neural network model for supervised learning consists of two phases, training phase and testing phase. Training samples for training phase are generated by the following rules: We first denote the messages bits of length N including K information bits **x** and N − K frozen bits as $\mathbf{x}'$, and then the transmitted codeword **u** of length N can be obtained by following the encoding rules of polar codes $\mathbf{u} = \mathbf{x}'\mathbf{G}$, where $\mathbf{G} = \boldsymbol{F}^{\otimes m}$. $\boldsymbol{F}^{\otimes m}$ is the m-th Kronecker power where $\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $m = log_2 N$. Finally, the received symbol vector **y** can be obtained when **u** passes through BPSK modulation and channel with AWGN.

Let $\mathbf{x} = [\mathrm{x}_0, \ldots \mathrm{x}_{K-1}]$, $\hat{\mathbf{x}} = [\hat{x}_0, \ldots \hat{x}_{K-1}]$ and $\mathbf{y} = [y_0, \ldots y_{N-1}]$. We define all possible **x** and $\hat{\mathbf{x}}$ by the set A and that of all possible **y** by the set B. Randomly picked a training sample **y** from B, there will be a corresponding label **x** in A. During training phase, we consider four factors that affect the performance of the RNN decoders.

- *Signal to noise ratio of training samples:* In the actual channel, the signal to noise ratio (SNR) is uncertain and time-varying, we simulate the noise in channel decoding phase to set different SNR of training samples and adopt the normalized validation error (NVE) in [4] to measure the performance of RNN decoders. The NVE defines as

$$\mathrm{NVE}(\rho_t) = \frac{1}{S}\sum_{s=1}^{S} \frac{BER_R(\rho_t, \rho_{v,s})}{BER_M(\rho_{v,s})} \tag{1}$$

  where ρ_t and ρ_v denote the SNR of training data sets and validation data sets, respectively. Let ρ_s be the s-th SNR in different validation data sets of the number S. $BER_R(\rho_t, \rho_{v,s})$ denotes BER achieved by RNN decoder, which trained at ρ_t on the data with $\rho_{v,s}$. Similarly, $BER_M(\rho_{v,s})$ is the BER achieved by MAP decoding at ρ_v . As such, our purpose of setting up different ρ_t is to find the optimal ρ_t which results in the least NVE in training phase, and then the optimal ρ_t will be used for the testing phase.

- *Ratio of codebook set A:* To evaluate the generalization ability of RNN decoders, we choose the information bits **x** which covers only part of A, that is, the ratio p of codebook set A. In the sequel, we set the ratio $p = 40\%, 60\%, 80\%, 100\%$, respectively.
- *Loss function:* Here, We choose the mean squared error (MSE) as the loss function, defined by

$$\mathrm{L_{MSE}} = \frac{1}{\mathrm{K}}\sum_{\mathrm{i}=0}^{\mathrm{K}-1}(\mathrm{x_i} - \hat{\mathrm{x}}_\mathrm{i})^2 \quad (2)$$

  where $\mathrm{x_i}$ is the label and $\hat{\mathrm{x}}_\mathrm{i}$ donates the i-th estimated information bit of RNN decoder.
- *Dropout:* There exists a saturation length for RNN decoder. When the saturation length exceeds codeword length N and $p < 100\%$, the RNN decoder will be overfitting. When the saturation length is under the codeword length N, the RNN decoder will suffer from the problem of underfitting no matter what the p is. Based on this, we compare the BER performance of the decoders with dropout and without dropout, instead of adding only dropout like [5].

3. Proposed RNN decoders

In this section, we describe three types of proposed RNN decoders, which are built upon LSTM, GRU and Bi-GRU, respectively.

3.1. LSTM decoder

In LSTM, gating mechanism is used to control the information flow such that the gradient vanishing problem in RNN is better handled, and the long range dependency is better captured. The flow diagram of our proposed LSTM decoder is shown in figure 2, which mainly consists of two parts, a LSTM cell and a fully connected layer with sigmoid activation function. Let M denote the Mini-batch size. The input of the LSTM decoder is $\mathbf{y} = (\boldsymbol{y_0}, \boldsymbol{y_1}, \dots, \boldsymbol{y_{N-1}})$, where $\mathbf{y_t}$ $(t = 0,1, \dots, N-1)$ of length M is the input of LSTM cell at time *t*. The memory cell vector $\boldsymbol{c_{t-1}}$ and cell output $\boldsymbol{h_{t-1}}$ at time $t-1$ will affect the memory cell vector $\boldsymbol{c_t}$ and the cell output $\boldsymbol{h_t}$ at time *t*.

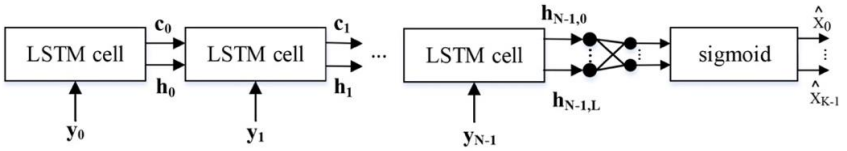

Figure 2. The flow diagram of LSTM decoder.

The LSTM cell in figure 3 includes input gate, forget gate and output gate. The input gate (output gate) uses inputs from other memory cell to decide whether to store (access) certain information in its memory cell. The forget gate determines the importance of memory cell vector $\boldsymbol{c_{t-1}}$ to the memory cell vector $\boldsymbol{c_t}$. The memory cell vector $\boldsymbol{c_t}$ is updated by adding a new memory content vector $\hat{\boldsymbol{c}}_\boldsymbol{t}$ and partially forgetting the existing memory, where $\hat{\boldsymbol{c}}_\boldsymbol{t}$ describes the state of current LSTM cell input. The value of the $\boldsymbol{i_t}$, $\boldsymbol{f_t}$, $\boldsymbol{o_t}$, $\hat{\boldsymbol{c}}_\boldsymbol{t}$, $\boldsymbol{c_t}$ and $\boldsymbol{h_t}$ are computed by

$$\boldsymbol{i_t} = \boldsymbol{\sigma}\left(\boldsymbol{W_{yi}}\boldsymbol{y_t} + \boldsymbol{W_{hi}}\boldsymbol{h_{t-1}} + \boldsymbol{b_i}\right) \quad (3)$$

$$\boldsymbol{f_t} = \sigma\left(\boldsymbol{W_{yf}}\boldsymbol{y_t} + \boldsymbol{W_{hf}}\boldsymbol{h_{t-1}} + \boldsymbol{b_f}\right) \quad (4)$$

$$\boldsymbol{o_t} = \sigma\left(\boldsymbol{W_{yo}}\boldsymbol{y_t} + \boldsymbol{W_{ho}}\boldsymbol{h_{t-1}} + \boldsymbol{b_o}\right) \quad (5)$$

$$\hat{\boldsymbol{c}}_\boldsymbol{t} = \tanh\left(\boldsymbol{W_{yc}}\boldsymbol{y_t} + \boldsymbol{W_{hc}}\boldsymbol{h_{t-1}} + \boldsymbol{b_c}\right) \quad (6)$$

$$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \hat{\boldsymbol{c}}_\boldsymbol{t} \quad (7)$$

$$\mathbf{h_t} = \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t}) \quad (8)$$

where $\sigma(\mathrm{x}) = \frac{1}{1+\mathrm{e}^{-\mathrm{x}}}$. The initial value $\boldsymbol{c_0} = \mathbf{0}$, $\boldsymbol{h_0} = \mathbf{0}$, where the $\mathbf{0}$ represents zero vector of length L. $\odot$ stands for the element-wise product of two matrices or vectors. The hidden state in LSTM cell is defined as the concatenation of ($\boldsymbol{h_t}$, $\boldsymbol{c_t}$), where the 'slow' state $\boldsymbol{c_t}$ fights the vanishing gradient problem and the 'fast' state $\boldsymbol{h_t}$ makes hard decision over short periods of time. Note that

final output of LSTM cell is $\boldsymbol{h_N}$, which is determined by the output gate $\boldsymbol{o_N}$ and the memory cell vector $\mathbf{c_N}$. Finally, we add a layer of sigmoid activation function to get the decoded value $\hat{\mathbf{x}}$ of length K by

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}\boldsymbol{h_N} + \boldsymbol{b}) \tag{9}$$

Our goal is to train a set of parameter $\{\boldsymbol{W_{yi}}, \boldsymbol{W_{hi}}, \boldsymbol{W_{yf}}, \boldsymbol{W_{hf}}, \boldsymbol{W_{yo}}, \boldsymbol{W_{ho}}, \boldsymbol{W_{yc}}, \boldsymbol{W_{hc}}, \mathbf{W}, \boldsymbol{b_i}, \boldsymbol{b_f}, \boldsymbol{b_o}, \boldsymbol{b_c}, \boldsymbol{b}\}$ to achieve the output $\hat{\mathbf{x}}$ which is as close as possible to the label **x**.

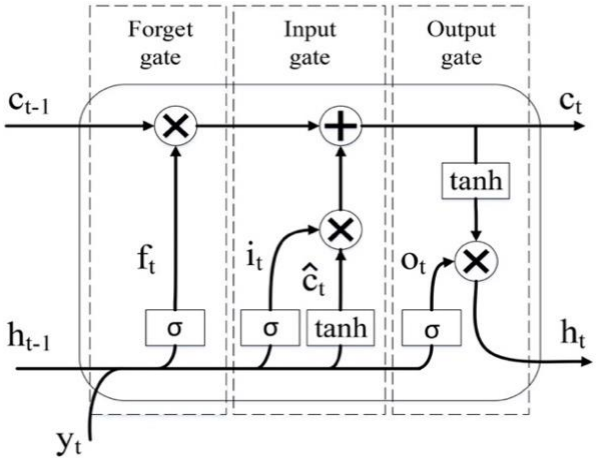

Figure 3. The LSTM cell

3.2. GRU decoder

Compared to LSTM decoder, the structure of GRU decoder is much simpler. GRU decoder and LSTM decoder both have gating mechanism to make the flow of information inside the unit. There is no separate memory cells in the GRU, which is diverse from the LSTM. The flow diagram of the proposed GRU decoder is shown in figure 4, which mainly consists of two parts, a GRU cell and a fully connected layer with sigmoid activation function. The input of the GRU decoder is also $\mathbf{y} = (\boldsymbol{y_0}, \boldsymbol{y_1}, \dots, \boldsymbol{y_{N-1}})$. The hidden state $\boldsymbol{h_{t-1}}$ at time $t-1$ will affect the state of GRU cell at time *t*.

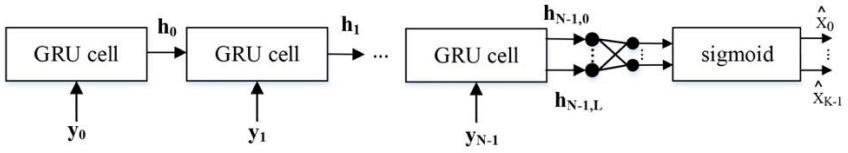

Figure 4. The flow diagram of GRU decoder.

The GRU cell in figure 5 includes two types of gates: update gate and reset gate. The hidden state $\boldsymbol{h_t}$ is define as a linear interpolation between previous hidden state $\boldsymbol{h_{t-1}}$ and the candidate hidden state $\widehat{\boldsymbol{h}}_t$ at time *t*. The formula is as follows:

$$\mathbf{h_t} = (\boldsymbol{I} - \boldsymbol{z_t}) \odot \boldsymbol{h_{t-1}} + \boldsymbol{z_t} \odot \widehat{\boldsymbol{h}}_t \tag{10}$$

where $\mathbf{I} = (1,1,\dots,1)$, whose dimension is L. An update gate is used to control how much the unit updates its hidden state. We compute $\boldsymbol{z_t}$ and the candidate hidden state $\widehat{\boldsymbol{h}}_t$ by

$$\mathbf{z_t} = \sigma(\boldsymbol{W_{hz}}\boldsymbol{h_{z-1}} + \boldsymbol{W_{yz}}\boldsymbol{y_t} + \boldsymbol{b_z}) \tag{11}$$

$$\widehat{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_y}\boldsymbol{y_t} + \boldsymbol{W_h}(\boldsymbol{r_t} \odot \boldsymbol{h_{t-1}}) + \boldsymbol{b_h}) \tag{12}$$

where $\boldsymbol{r_t}$ decides how much the unit ignores its candidate hidden state $\widehat{\boldsymbol{h}}_t$. The smaller the value of the $\boldsymbol{r_t}$, the more ignored. $\boldsymbol{r_t}$ is computed similarly to the $\mathbf{z_t}$:

$$\boldsymbol{r_t} = \sigma(\boldsymbol{W_{hr}}\boldsymbol{h_{t-1}} + \boldsymbol{W_{yr}}\boldsymbol{y_t} + \boldsymbol{b_r}) \tag{13}$$

We also add a layer with full connection in the end, same with (9).

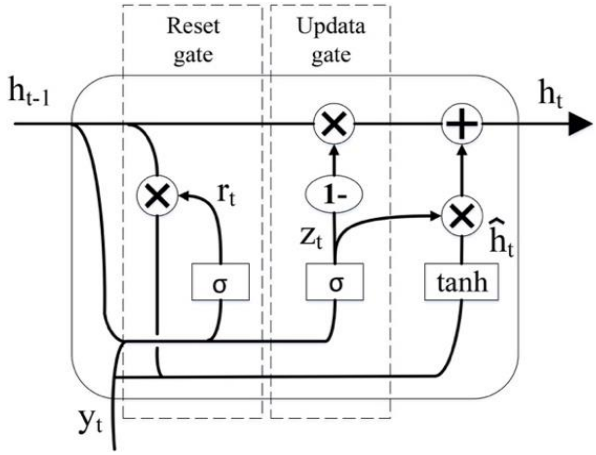

Figure 5. The GRU cell.

3.3. Bi-GRU decoder

Bi-GRU decoder is proposed to be able to exceed the LSTM decoder in BER performance. Since the GRU decoder only models the information flow in one direction, we use a Bi-GRU decoder to get information by summarizing from both directions. More specifically, one GRU goes from 1 to N and the other from N to 1 if given a sequence of length N.

As shown in figure 6, the structure of Bi-GRU is the combination of two different directional GRUs. At each time *t*, we provide both directional GRU the same input, which is same the input of GRU, and the output is determined by both two GRUs. Denote the hidden states of the forward and backward GRUs as $\overrightarrow{\boldsymbol{h_t}}$ and $\overleftarrow{\boldsymbol{h_t}}$ at time t, respectively. The cell output is computed by concatenating the two hidden states at each time. $\overrightarrow{\boldsymbol{h_t}}$ and $\overleftarrow{\boldsymbol{h_t}}$ are computed by (10)-(12), now we simply replace the above complex calculation with the function f:

$$\overrightarrow{\mathbf{h_t}} = \mathrm{f}\left(\mathbf{W_1 y_t} + \mathbf{W_2}\overrightarrow{\mathbf{h_{t-1}}} + \vec{\mathbf{b}}\right) \tag{14}$$

$$\overleftarrow{\mathbf{h_t}} = \mathrm{f}\left(\mathbf{W_3 y_t} + \mathbf{W_4}\overleftarrow{\mathbf{h_{t-1}}} + \overleftarrow{\mathbf{b}}\right) \tag{15}$$

$$\mathbf{o_t} = \left(\overrightarrow{\mathbf{h_t}}, \overleftarrow{\mathbf{h_t}}\right) \tag{16}$$

where the representation in (16) means to concatenate the forward and backward outputs. As before, we add a layer of with full connection as the last layer to obtain the decoded value of Bi-GRU decoder. The decoded value $\hat{\mathbf{x}}$ is computed by

$$\hat{\mathbf{x}} = \sigma(\boldsymbol{W'o_N} + \boldsymbol{b'}) \tag{17}$$

where N is also the total time and the dimension of $\hat{\mathbf{x}}$ is also K.

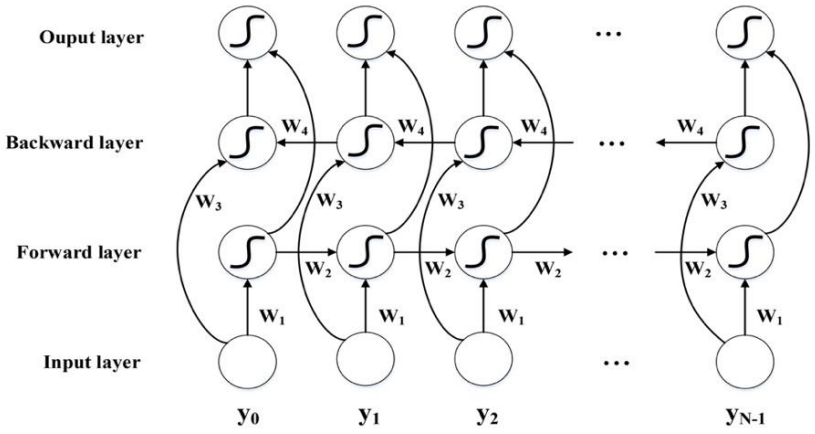

Figure 6. The architecture of Bi-GRU.

4. Performance evaluation

We compare the performance of proposed three RNN decoders with LSTM, GRU and Bi-GRU of code length N=8, 32, respectively. We focus on the polar code of rate 1/2 for performance evaluation. In addition, in order to reduce the decoding complexity, we only use a single layer RNN as decoder. For more parameter settings, we select hyperparameters set, which is shown in table 1, for RNN decoders after many trials.

We first observe the computational time of these three decoders in figure 7. The backward propagation time for one training sample of LSTM decoder is almost twice that of the GRU decoder, and Bi-GRU decoder is about three times that of GRU decoder under the same N. For the forward propagation time for one testing sample under the same N, the LSTM decoder is almost 21 times longer than GRU decoder, and Bi-GRU decoder is twice as long as LSTM decoder. Therefore, the GRU decoder is the most time saving in computing.

Table 1. Hyperparameters setting

Item	Hyperparameters setting
Number of training samples	10^6
Number of testing samples	10^5
Total training epoch	10^5
Mini-batch size	128
Optimization method	Adam optimization
SNRs for training (ρ_t)	{2,0,2,4,6,8,10,12,14,16,18,20}dB
SNRs for testing (ρ_v)	{0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6}dB
Training ratio of codebook p	40%,60%,80%,100%
Initialization method	Xavier initialization
L	256

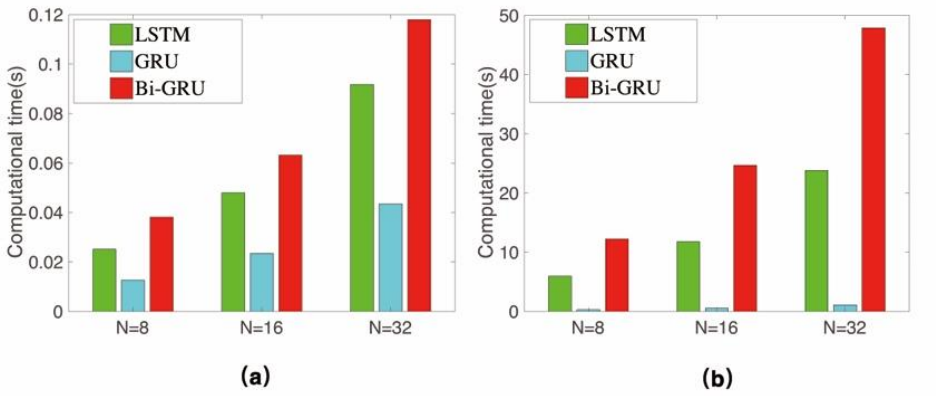

Figure 7. The computational time of LSTM, GRU and Bi-GRU under different length of codeword N with noise. (a) The time for one training sample through backward propagation. (b) The time for one testing sample through forward propagation.

Figure 8. shows the case when $N = 8$ with noise. We can see that all three RNN decoders can achieve MAP performance no matter dropout or not when p = 100%. When p < 100%, the BER performance of Bi-GRU decoder is significantly better than LSTM decoder and GRU decoder although they are all overfitting, and three RNN decoders with dropout are better than decoders without dropout.

Figure 9. shows the case when N = 32 with noise. We can see that as p increases, the overall trend of BER for all RNN decoders increases, which can be interpreted as they are all underfitting. The Bi-GRU decoder always has a slight advantage in BER performance than LSTM decoder and GRU decoder under different p. The three RNN decoders without dropout are always better than the one with dropout when $p = 40\%, 60\%, 80\%$. In figure (d), the LSTM decoder without dropout has better BER performance than LSTM decoder with dropout and the GRU decoder without dropout has better BER performance than GRU decoder with dropout. But the Bi-GRU decoder with dropout makes a breakthrough in BER performance, which is even better that Bi-GRU without dropout, this also shows that Bi-GRU is prone to overfitting.

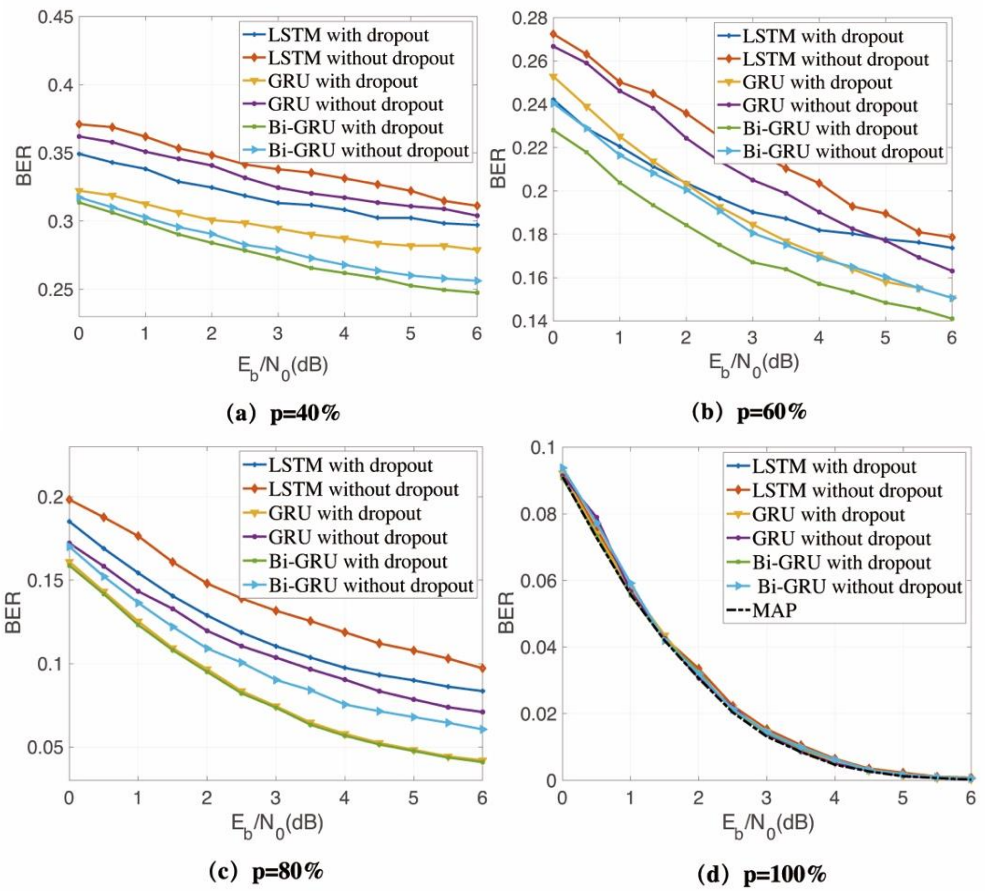

Figure 8. The BER performance achieved by LSTM, GRU and Bi-GRU when $N = 8$ under different training ratio $p = 40\%, 60\%, 80\%$ and 100%.

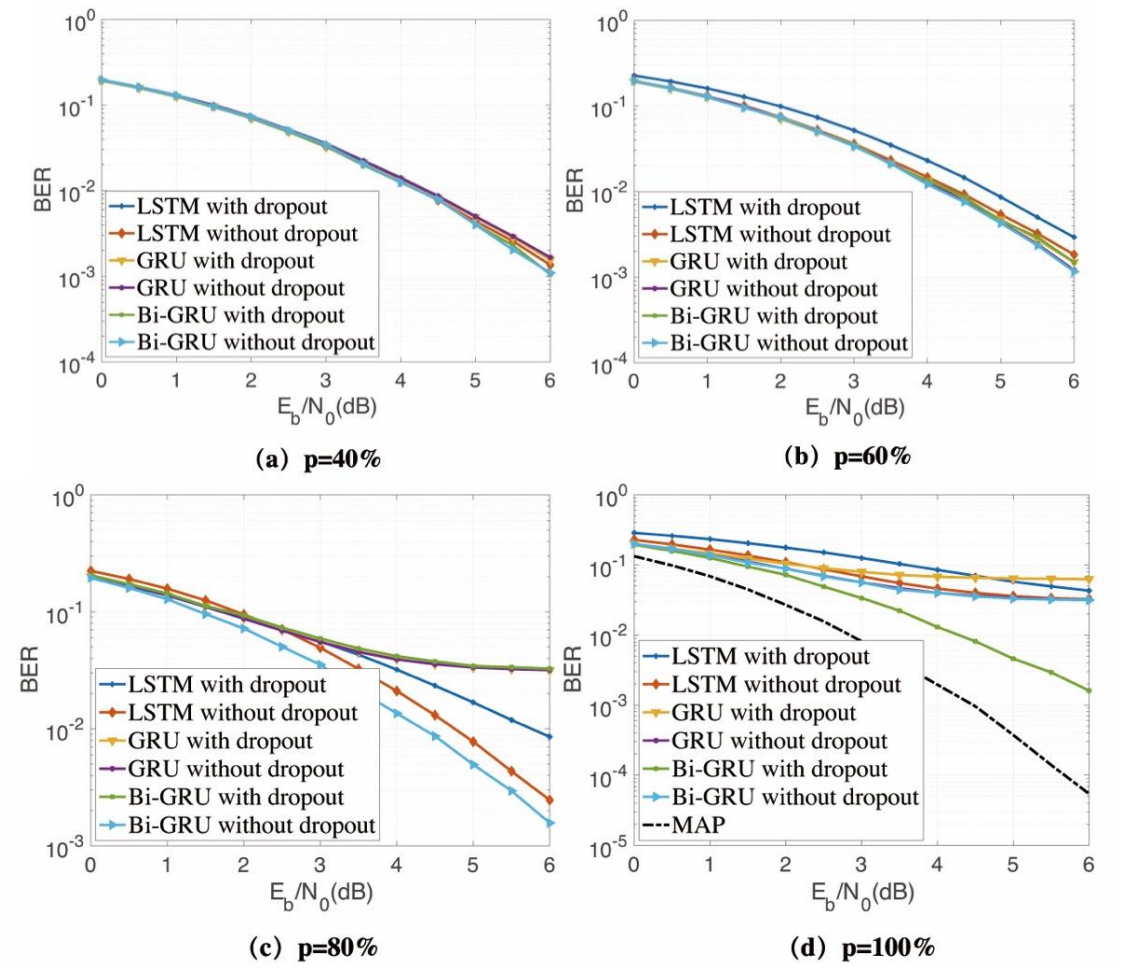

Figure 9. The BER performance achieved by LSTM, GRU and Bi-GRU when $N = 32$ under different training ratio $p = 40\%, 60\%, 80\%$ and 100%.

5. Conclusion

In this paper, we proposed three types of RNN decoders, named LSTM decoder, GRU decoder and Bi-GRU decoder, respectively. After extensive experiments, we found that the BER performance of GRU decoder, which had a simpler structure and less computational time, was not worse than the LSTM decoder. Although the Bi-GRU decoder always had the best BER performance, it was at the expense of more computational time and was prone to overfitting. We confirmed that the performance of RNN decoders with dropout was better than the decoders without dropout when decoding models were overfitting, while the RNN decoders without dropout was better than decoders with dropout when decoding models were underfitting.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61671143).

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, pp. 770-778, 2016.

[2] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention based neural machine translation," *Conference on Empirical Methods in Natural Language Process*, Lisbon, Portugal, pp. 1412-1421, 2015.
[3] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," *IEEE International Conference Acoustics, speech and signal Process (ICASSP)*, Vancouver, Canada, pp. 6645- 6649, 2013.
[4] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning- based channel decoding," in *The 51st Annual Conference on Information Sciences and Systems (CISS). IEEE*, Nicollet MallMinneapolis, pp. 1- 6, 2017.
[5] W. Lyu, Z. Zhang, C. Jiao, et al. "Performance Evaluation of Channel Decoding With Deep Neural Networks," in *IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, pp. 1-6, 2018.