A Convolutional Neural Network Decoder for Convolutional Codes

Zhengyu Zhang[1(✉)], Dongping Yao[1], Lei Xiong[1], Bo Ai[1], and Shuo Guo[2]

[1] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China
18120185@bjtu.edu.cn

[2] Shanghai Gezhi High School, Shanghai 200001, China

Abstract. The convolutional neural network (CNN) decoder for general convolutional decoding is proposed. The parameters of CNN are determined by the initial state of each input block and the constraint relationship between adjacent bits is extracted by the convolutional layer as the constraint features. Then CNN decoder realizes decoding process through the extracted constraint feature instead of codewords directly. The result shows that, without changing the structure of decoder, the decoding performance of CNN decoder on different convolutional codes is equivalent to Viterbi soft decoding algorithm. Compared with Viterbi decoding, the larger constraint length or the lower SNR, the greater gain can be obtained in CNN decoder. Besides, we consider CNN trained by the two kinds of training sets in order to further investigate the potential and limitations of CNN decoder with respect to decoding performance, analysing the advantages and factors of these two kinds of training sets.

Keywords: Deep learning · Convolutional code · Viterbi decoding algorithm · Neural network

1 Introduction

Convolutional codes are linear codes with a very distinct algebraic structure, which assign code bits to an incoming information bit stream continuously in a stream-oriented fashion. As an important channel coding technology, convolutional codes have been widely used in communication systems such as GSM, WCDMA, CDMA2000, and broadcasting.

In recent years, deep learning methods have developed rapidly and applied to all layers of communication systems to greatly improve the performance, including channel decoding. In [1] the author revisits the idea of using deep neural networks for one-shot decoding of random and structured codes. It provides that neural networks can learn a form of decoding algorithm, rather than a simple classifier. In [2] the author proposes a deep learning method for improving the belief propagation algorithm. The method generalizes the standard belief propagation algorithm by assigning weights to the edges of the Tanner graph.

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 313, pp. 113–125, 2020.
https://doi.org/10.1007/978-3-030-41117-6_10

In [3] it introduces a recurrent neural decoder architecture based on the method of successive relaxation. The results demonstrate that the neural belief propagation decoder can be used to improve the performance, or alternatively reduce the computational complexity, of a close to optimal decoder of short BCH codes. In [4] it shows that the conventional iterative decoding algorithm for polar codes can be enhanced when sub-blocks of the decoder are replaced by neural network (NN) based components. In [5] the author proposes an iterative belief propagation convolutional neural network (BP-CNN) architecture for channel decoding. The results show that this architecture Iterating between BP and CNN can improve the decoding signal to noise ratio (SNR) and result in better decoding performance. [6] and [7] discuss an improved Viterbi soft decision method based on artificial neural network, which can improve the efficiency of decision making in comparison with other methods of decision making, in term of bit error rate (BER). The [8–10] discuss the structure of convolutional code decoder based on recurrent neural network. The results show that the neural network decoder has lower complexity and better performance than Viterbi decoding.

However, most of the current research about convolutional decoding focus on the a certain convolutional code, which cannot adapt to different convolutional codes. With the rapid development of mobile communication, different communication systems use different convolutional coding schemes (such as different code rates and constraint length). The traditional decoder, such as Viterbi decoder, needs to be designed separately, which seriously increases the design difficulty and cost at the terminal. In order to solve these problems, it is urgent to consider a convolutional decoder with general architecture to autonomously adapt to different convolutional codes.

In this paper, we propose a general convolutional decoder based on neural network and analyze the decoding performance. Inspired by the receptive field of convolution kernels, we consider CNN to realize convolutional decoding process where the parameters of CNN decoder is determined by the initial state of each coding block, thus ensuring the integrity of adjacent bits between two coding block. Besides, the convolutional layer is adopted to learn the constraint relationship of the convolutional code, so the CNN decoder realizes decoding process on the basis of learned constraint features, instead of directly the received sequence. The results show that the CNN decoder can adapt to different convolutional codes without changing the structure, and decoding performance is better than Viterbi soft decoding. Besides, we analyse the advantages and factors of received-sequence training sets and error-pattern training sets.

The rest of this paper is organized as follows. Section 2 proposes the scheme of decoder, which is carried out from four aspects: system model, training set generation, decoding process and convolutional neural network design. Section 3 presents the performance comparison of the CNN decoder trained by different training sets and simulation about general decoding capability. Conclusions are drawn in Sect. 4.

2 Neural Network Decoder

This section introduces the neural network decoding system model, training set generation, decoding process and the CNN decoder structure design. The principle of decoding process and implementation of CNN decoder are also explained in this section.

2.1 System Model

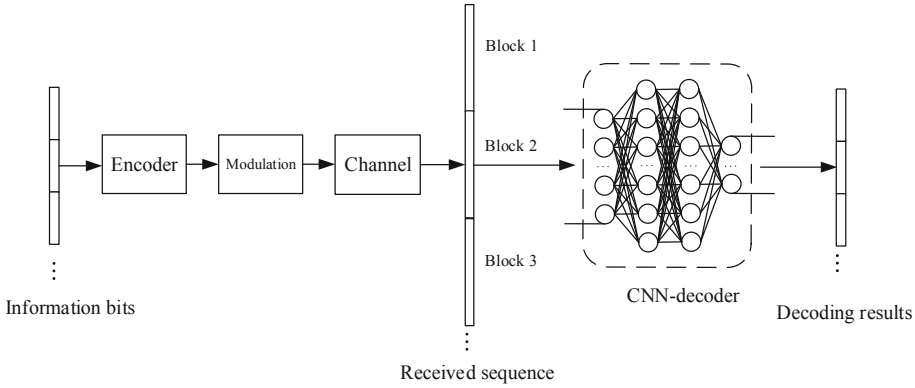

Fig. 1. System model.

This section introduces the system model from the perspective of neural networks. System model is shown as Fig. 1. The information bits sequence m is sent to the receiver via the encoder, modulation and channel. At receiver, the decoding results can be expressed as:

$$\hat{m} = D(R) \tag{1}$$

where R denotes the received sequence, $D(\cdot)$ denotes mapping of the decoder, and $\hat{m}$ denotes the output of the decoder. In the neural network decoder, decoding mapping is learned from the training sets, which is stored as the weight parameters in the neural network. The received sequence R is segmented into blocks of fixed length that decided by the window size of decoder. One block denoted as $\boldsymbol{r} = [r_1, r_2, \ldots, r_k]^T$ is an input vector for neural network, where k is the window size of decoder. For neural nodes in the input layer, the net input, written as v, representing the weighted sum of the input vector, can be expressed as:

$$v = \sum_{i=1}^{k} w_i r_i + b = \boldsymbol{w}^T \boldsymbol{r} + b \tag{2}$$

where $\boldsymbol{w} = [w_1, w_2, \ldots, w_k]^T$ is a k-dimensional weight vector, and b denotes bias. The activation value of the neural node is generated via passing v through the activation function $f(\cdot)$. In order to learn the complex decoding mapping, the neural network decoder connects many neural nodes by a certain hierarchical structure. Given the parameter of the neural network decoder is θ, the decoding mapping of neural network can be expressed as:

$$\hat{m} = f(R;\theta) = f^{(L-1)}(f^{(L-2)}(...f^{(0)}(R))) \tag{3}$$

where L is the number of layers in the neural network, and is also called *depth*.

2.2 Training Set

This section presents the generation of two kinds of training sets. The neural network decoder learns decoding mapping from the training sets and the training sets decide the upper limit of the decoding capability. The reasonable selection and processing of the training set, for example adding error patterns, can enhance the decoding capability of the neural network. Besides, the neural network decoder can learn error correction capability from reasonable training set to correct errors caused by transmission. For convolutional decoding, there are two kinds of training sets: the error-pattern training set and the received-sequence training set. The generation of them is shown in Fig. 2.

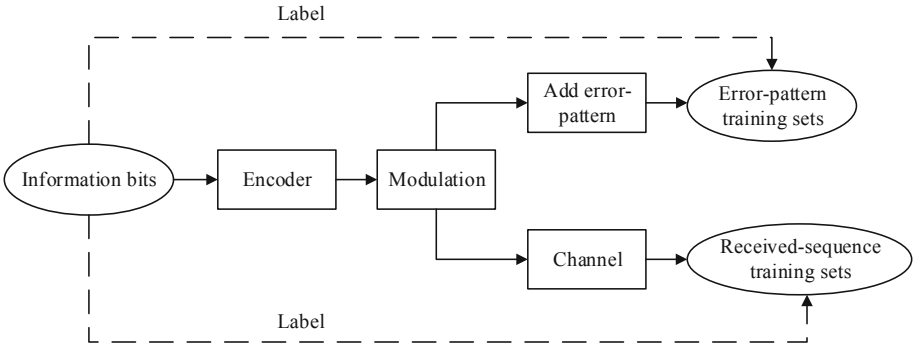

Fig. 2. The generation of the error-pattern training sets and the received-sequence training sets.

The generation of the error-pattern training set is described as follows. Suppose the length of information bits is p, then there are 2^p kinds of codewords without errors, and the length of codeword is q ($q > p$). In the *1*-error pattern training set, we consider all cases where only one bit error occurs in each block and there are $2^p \times (C_q^0 + C_q^1)$ kinds of samples in this training set. In the *2*-errors pattern training set, we consider all cases where one bit or two bits error occur in each block, and there are $2^p \times (C_q^0 + C_q^1 + C_q^2)$ samples in this training set. In the

r-errors pattern training set, the size of the training set is $2^p \times (\sum_{i=0}^{r} C_q^i)$. Each encoded block is used as the input of the neural network, and the corresponding uncoded information bits are used as the label.

The generation of the received-sequence training set is described as follows. Each encoded block impacted by the channel is used as the input of the neural network, and the corresponding uncoded information bits are used as the label.

The error-pattern training set considers all the correct cases and error cases, and the difference between the error samples and correct samples is obvious and clear. The learned decoding capability can reach to higher upper limit. In the received-sequence training set, samples are labeled vaguely, where failing to label correct and error samples clearly. But the generation of the received-sequence training set does not require additional computational cost, which can reduce the training cost. The neural network interface trained by these two training sets is shown in Fig. 3, the color depth represents the probability of decoding results, yellow means the decoding result is *1* and blue means the decoding result is *0*.

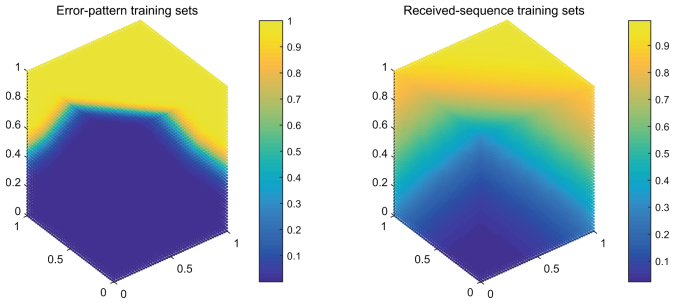

Fig. 3. Neural network interface trained by the error-pattern training sets (left) and the received-sequence training sets (right). (Color figure online)

2.3 Decoding Process

This section presents the decoding process of the neural network decoder. In the process of convolutional code encoding, due to the existence of the shift register, the convolutional code has strong constrained relationship between adjacent bits. If the continuous convolutional code is directly grouped into the neural network, the first and last bits of each block will lose the constrained information, resulting in a very poor decoding capability. Therefore, decoding process is required to realize the integrity of the input information between adjacent bits during convolutional decoding.

The decoding process is shown in Fig. 4. Taking the (2,1,3) convolutional code as the example, the size of shift registers in the encoder is 2, and there are four initial states 00, 01, 10, and 11 during encoding. When encoding, the initial state of the block is decided by the last two bits of the previous block. We train various neural network models corresponding the different initial states.

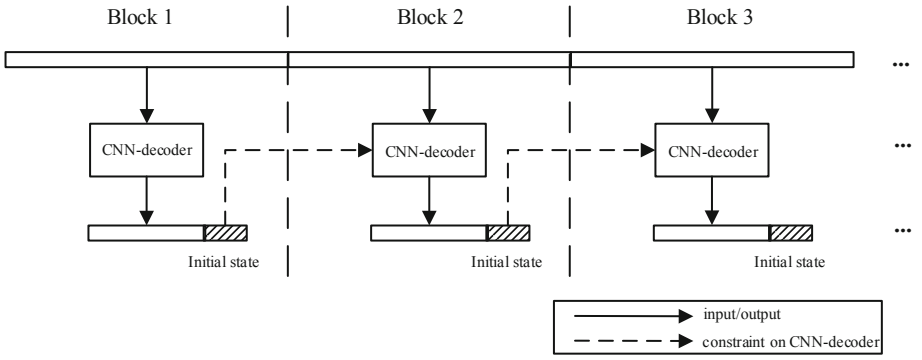

Fig. 4. The decoding process.

In (2,1,3) convolutional code, there are four neural network models corresponding four initial states. At the decoding process, we select the reasonable neural network model according to the previous decoding result and determine this model to complete convolutional decoding. Then repeat these operation in the next decoding block. These operation can ensure the integrity of the input information between adjacent bits of different blocks, and can also ensure the continuity of the decoding result without repeated decoding or missing decoding. Besides, every neural network model only corresponds to an initial state, thus the learning pressure for decoding mapping can be reduced, avoiding the neural network model being too complicated and redundant.

2.4 Convolutional Neural Network Design

This section presents the advantage of the CNN in convolutional decoding and the design of the CNN decoder. During the decoding process, the feature of codeword is firstly extracted by the convolutional layer, which is affected by the constraint relationship, named *Constraint feature*. Then CNN learns the decoding mapping from the extracted constraint feature. Compared with learning from codewords directly, it can ensure the integrity of continuous information preferably and does not cut the constraint relationship between adjacent bits.

In convolutional code, there is a certain constraint relationship between adjacent bits because of the shift register. The decoding process is equivalent to return the codeword to original information bits through the constraint relationship. Figure 5 shows the constraint feature of the (2,1,3) convolutional code. The bit that affects a batch of codewords is named as the *key bit* of corresponding sequence. In Fig. 5, the constraint length is 3, the key bit affects 6 continuous bits of codewords at most and we regard these 6 bits as the receptive field of the convolution kernel. The convolution kernel can learn the mapping between the key bit and the codeword. For decoding, the convolution kernel extracts the constraint feature from codewords via a certain receptive field range, and the constraint feature determines the value of the key bit, outputting as the decoding result.

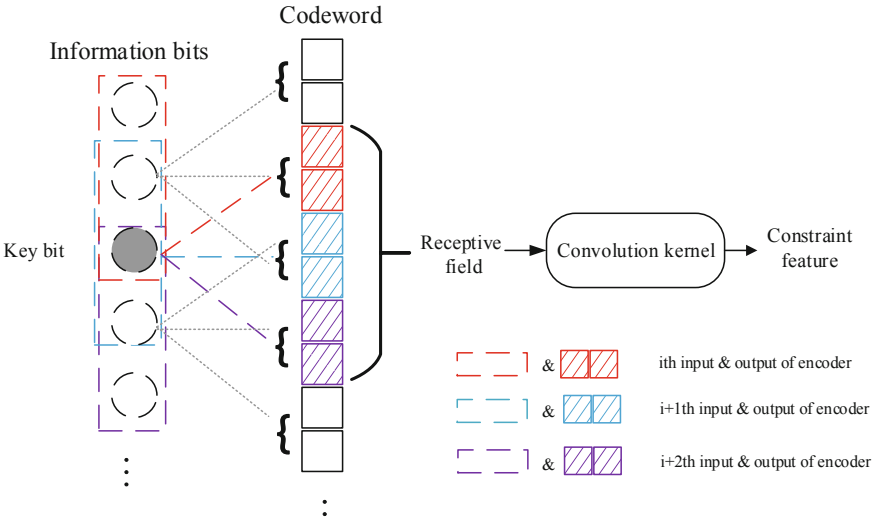

Fig. 5. The constraint feature learned from the codeword by the convolutional kernel.

The structure of CNN decoder is designed as Fig. 6, including an input layer, a convolutional layer, a flatten layer, a fully connected layer and an output layer. The size of input vector is 1×24, i.e., the window size of CNN decoder is 24. In the convolutional layer, the convolution kernel realizes feature-extraction by one-dimensional convolutional operation with size 1×10 and number 150. Feature maps, with size 1×15 and number 150, can be obtained via the convolutional layer. The single feature map represents the learned constraint feature. The flatten layer converts the multidimensional feature maps to one-dimensional. Through the fully connected layer and output layer, the size of vector changes to 1×12, which represents the decoding results. In addition, the RMSProp (root mean square prop) algorithm is used to update the parameters to accelerate the convergence. The Dropout and Early-Stopping strategies are adopted to avoid over-fitting problems.

3 Performance

This section compares the training results of different training sets and simulates the general decoding capability of the CNN decoder for different convolutional codes. The measure of decoding performance is BER.

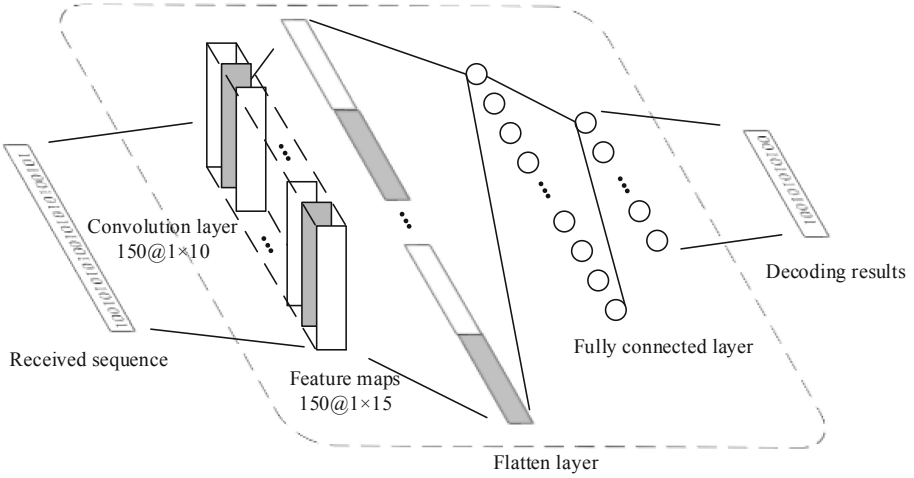

Fig. 6. The structure of CNN decoder.

3.1 Parameter Setting

In the simulation, we consider the error-pattern training sets including *1*-error, *2*-errors, *3*-errors and the received-sequence training sets affected by the SNR from 2 dB to 7 dB. The specific simulation parameters are shown in Table 1.

3.2 Simulation Result

This section gives the comparison of different training sets based on (2,1,3) convolutional code and the decoding performance of CNN decoders on (2,1,3), (2,1,5), (2,1,7), (2,1,9) convolutional codes, which are compared with the Viterbi decoding algorithm. Besides, the influence of constraint lengths on the decoding capability is considered.

Figure 7 shows the performance of the decoder trained by received-sequence sets with different SNR and Fig. 8 shows the performance of the decoder trained by error-pattern training sets with different error numbers. The results show that the received-sequence training sets with lower SNR can lead to better performance of CNN decoder and too many errors lead to confusion in the error-pattern training set, resulting in the poor performance of CNN decoder. For received-sequence training sets, there are less errors of received sequence during the transmission with higher SNR, therefore the error correction capability learned from the training sets will be weaker, causing the higher BER. For convolutional codes, because of the fixed constraint length, their error correction capability is limited. Too many errors have exceeded the error correction capability and lead to confusion in the training sets, resulting in the poor performance. For (2,1,3) code, the CNN decoder trained by *2*-errors pattern sets has better decoding performance.

Table 1. Simulation parameter

Parameters	Value
Training set	Error-pattern training set Received-sequence training set
Training set size[a]	100,000
Test set size	1,000,000
Modulation	BPSK
Channel	AWGN
Convolutional code	(2,1,3), (2,1,5), (2,1,7), (2,1,9)
Generator polynomial	[5,7], [23,33], [133,65], [753,561]
Decoder window size	24
Convolution kernel size	1×10
Convolution kernel number	150
Fully connected neural nodes	1,000
Decoder output size	12
Batch-size	1,000
Dropout rate	0.3
Early-Stopping steps	2

[a]Only list the received-sequence training set size. The error-pattern training set size depends on the number of errors.

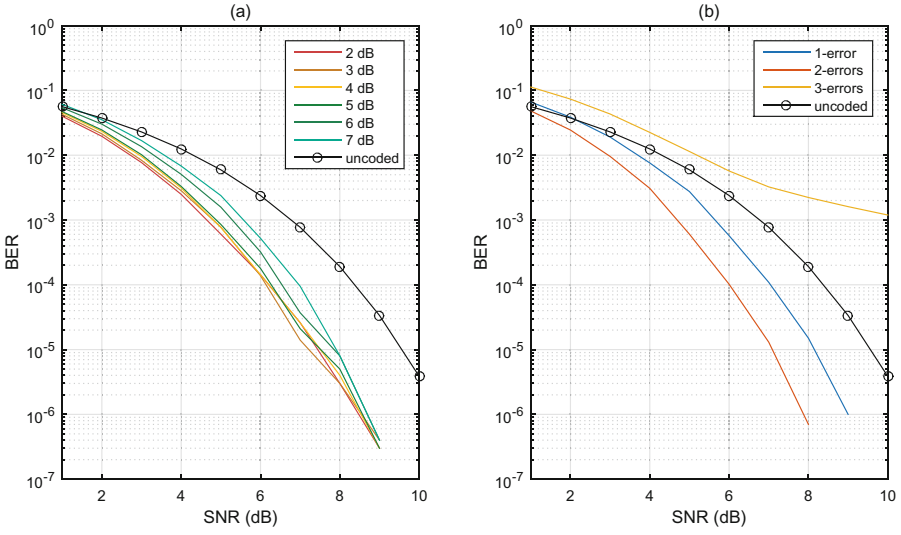

Fig. 7. Decoding performance of the CNN decoder trained by received-sequence training sets (a) and error-pattern training sets (b).

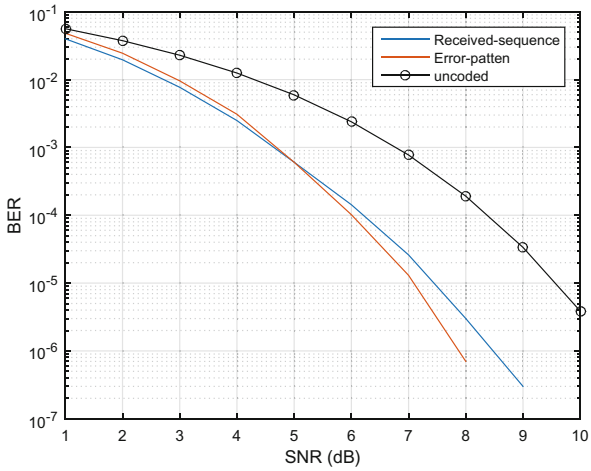

Fig. 8. Decoding performance comparison of the two training sets.

The comparison of the two kinds of training sets is shown as Fig. 8, where the 2 dB received-sequence training set and the *2*-errors pattern training set are selected as representative, which perform better based on the previous analysis. The results show that the received-sequence training sets perform better in lower SNR environment and the error-pattern training sets perform better in higher SNR environment. It is caused by the interface of trained CNN decoder. The interface trained by the received-sequence training set is vague, bringing better noise tolerance capability in low SNR environment. The interface trained by the error-pattern training set is clear, bringing the more accurate decoding result in high SNR environment. In order to analyze the upper limit of the decoding performance, the error-pattern training set is considered in the subsequent simulation.

Figure 9 shows the BER of the (2,1,3), (2,1,5), (2,1,7) and (2,1,9) convolutional decoding. The results prove that the CNN decoder can learn general decoding capability, performing better than Viterbi decoding algorithm for different convolutional codes. Besides, the longer constraint length or lower SNR, the greater degree of improvement can be obtained on CNN decoder. In order to compare easily, we set the each block on the initial state of all-zero, and ignore these known bits information when calculating BER. For (2,1,3) code, the decoding performance of CNN decoder trained by *2*-errors pattern training set is between Viterbi hard and soft decoding algorithm. It should be noted that the performance of CNN decoder trained by *3*-errors pattern training set is poor because of the limited error correction

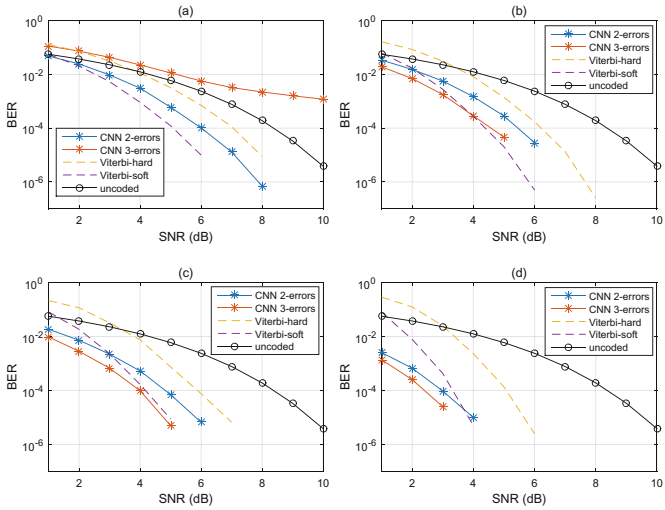

Fig. 9. Decoding performance for (2,1,3) Convolutional Code (a), (2,1,5) Convolutional Code (b), (2,1,7) Convolutional Code (c) and (2,1,9) Convolutional Code (d).

capability in (2,1,3) code. The performance of CNN decoder trained by the *2*-errors pattern training set is between Viterbi hard and soft decoding algorithm. For (2,1,5) code, the decoding performance of CNN decoder trained by *3*-errors pattern training set is better than Viterbi soft decoding when SNR < 4 dB, and slightly worse than Viterbi soft decoding when SNR > 4 dB. The performance of CNN decoder trained by the *2*-errors pattern training set is between Viterbi hard and soft decoding algorithm. For (2,1,7) code, the CNN decoder trained by the *3*-errors pattern training set can obtain the better performance all over SNR. About 0.2 dB gain can be obtained compared to Viterbi soft decoding algorithm when BER $= 10^{-4}$. The CNN decoder trained by *2*-errors pattern training set performs better than Viterbi soft decoding algorithm when SNR < 3 dB. For (2,1,9) code, the degree of improvement on CNN decoder trained by the *3*-errors pattern training set becomes greater. We can obtain gain of about 1 dB compared to Viterbi soft decoding when BER $= 10^{-4}$. And the CNN decoder trained by the *2*-errors pattern training set performs better than Viterbi soft decoding algorithm when SNR < 3.5 dB. In summary, the CNN decoder has better general decoding capability for different convolutional codes. When the structure of decoder is fixed, CNN decoder can realize decoding of different convolutional codes without structure changed, and the decoding performance is equivalent to Viterbi soft decoding algorithm, even better. Especially with the constraint length long enough or under the low SNR environment, the performance gain of the CNN decoder is greater.

4 Conclusion

In our work, the CNN decoder for general convolutional decoding is proposed. The parameters of CNN are determined by the initial state of each input block and the constraint relationship between adjacent bits is extracted by the convolutional layer as the constraint features. Then CNN decoder realizes decoding process through the extracted constraint feature rather than codewords directly. We analyse the advantages and factors of two kinds of training sets, knowing that the CNN decoder trained by received-sequence training sets obtains better noise tolerance capability, performing better in low SNR, and the CNN decoder trained by error-pattern training sets obtains better decoding capability, performing better in high SNR. Besides, the simulation shows that the decoding performance on different convolutional codes is equivalent to Viterbi soft decoding algorithm without changing the structure of CNN decoder. Compared with Viterbi soft decoding, the larger constraint length or the lower SNR, the greater gain can be obtained in CNN decoder.

Acknowledgments. This work was supported by the National Key Research and Development Program under Grant 2016YFE0200900, the Fundamental Research Funds for the Central Universities (2018JBM079), Beijing Natural Haidian Joint Fund under Grant L172020, Major Projects of Beijing Municipal Science and Technology Commission under Grant Z181100003218010, and the Royal Society Newton Advanced Fellowship (Grant no. NA191006).

References

1. Gruber, T., Cammerer, S., Hoydis, J.: On deep learning-based channel decoding. In: 2017 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, pp. 1–6 (2017)
2. Nachmani, E., Be'ery, Y., Burshtein, D.: Learning to decode linear codes using deep learning. In: 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, pp. 341–346 (2016)
3. Nachmani, E., Marciano, E., Lugosch, L., Gross, W.J., Burshtein, D., Beery, Y.: Deep learning methods for improved decoding of linear codes. IEEE J. Sel. Top. Sig. Process. **12**(1), 119–131 (2018)
4. Cammerer, S., Gruber, T., Hoydis, J., ten Brink, S.: Scaling deep learning-based decoding of polar codes via partitioning. In: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, pp. 1–6 (2017)
5. Liang, F., Shen, C., Wu, F.: An iterative BP-CNN architecture for channel decoding. IEEE J. Sel. Top. Sig. Process. **12**(1), 144–159 (2018)
6. Berber, S.M.: Soft decision output decoding (SONNA) algorithm for convolutional codes based on artificial neural networks. In: 2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791), Varna, Bulgaria, vol. 2, pp. 530–534 (2004)
7. Charei, Q.N., Aghamalek, H.F., Razavi, S.M., Golestanian, M.: An improved soft decision method in Viterbi decoder using artificial neural networks. In: 2013 First Iranian Conference on Pattern Recognition and Image Analysis (PRIA), Birjand, pp. 1–4 (2013)

8. Hamalainen, A., Henriksson, J.: A recurrent neural decoder for convolutional codes. In: 1999 IEEE International Conference on Communications (Cat. No. 99CH36311), Vancouver, BC, vol. 2, pp. 1305–1309 (1999)
9. Hamalainen, A., Henriksson, J.: Convolutional decoding using recurrent neural networks. In: International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), IJCNN 1999, Washington, DC, USA, vol. 5, pp. 3323–3327 (1999)
10. Hueske, K., Gotze, J., Coersmeier, E.: Improving the performance of a recurrent neural network convolutional decoder. In: 2007 IEEE International Symposium on Signal Processing and Information Technology, Giza, pp. 889–893 (2007)