A Parallel Turbo Decoder Based on Recurrent Neural Network

Li Zhang
Xidian University
weihong fu (whfu@mail.xidian.edu.cn)
Xidian University
Fan Shi
Shenzhen GrenTech WL Communication Limited
Chunhua Zhou
Shanghai Radio Equipment Research Institute
Yongyuan Liu
Rayfond Technology CO Limited

Research Article

Keywords: turbo code, neural network, bit error rate, channel noise

Posted Date: November 1st, 2021

DOI: https://doi.org/10.21203/rs.3.rs-831836/v1

A Parallel Turbo Decoder based on Recurrent Neural Network

Li Zhang[1], Weihong Fu[1]*, Fan Shi[2], Chunhua Zhou[3], Yongyuan Liu[4]

[1] School of Telecommunications Engineering, Xidian University, Xi’an, 710071, Shaanxi, China

[2] Shenzhen GrenTech WL Communication Limited, Shenzhen, 518057, China

[3] Shanghai Radio Equipment Research Institute, Shanghai, 201109, China

[4] Rayfond Technology CO, Ltd, Beijing, 100094, China

Abstract: A neural network-based decoder, based on a long short-term memory (LSTM) network, is proposed to solve the problem of high decoding delay caused by the poor parallelism of existing decoding algorithms for turbo codes. The powerful parallel computing and feature learning ability of neural networks can reduce the decoding delay of turbo codes and bit error rates simultaneously. The proposed decoder refers to a unique component coding concept of turbo codes. First, each component decoder is designed based on an LSTM network. Next, each layer of the component decoder is trained, and the trained weights are loaded into the turbo code decoding neural network as initialization parameters. Then, the turbo code decoding network is trained end-to-end. Finally, a complete turbo decoder is realized. Simulation results show that the performance of the proposed decoder is improved by 0.5–1.5 dB compared with the traditional serial decoding algorithm in Gaussian white noise and t-distribution noise. Furthermore, the results demonstrate that the proposed decoder can be used in communication systems with various turbo codes and that it solves the problem of high delay in serial iterative decoding.

Keywords: turbo code; neural network; bit error rate; channel noise

1. Introduction

With the rapid advancement of information technology, signal types and quantity are increasing exponentially[1]. Traditional signal processing methods have difficulties meeting the needs of diversified, refined, and intelligent communication demands in the new era[2,3]. How to ensure the efficient and reliable transmission of communication systems is the research hotspot in the field of communication technology[4,5] Channel coding technology improves the reliability of information transmission by adding redundant bits to the information sequence[6,7]. Turbo code is an excellent channel coding method with low coding complexity, good error correction performance, and strong versatility, widely used in various fields[8,9]. The most commonly used decoding algorithms of turbo codes include maximum likelihood sequence detection[10], maximum a posteriori decoding algorithm (MAP)[11,12], and improved MAP algorithm, etc. The MAP algorithm can be implemented using Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithms[13], grid-graph-based algorithms to maximize the posterior probability of error-correcting codes. They use serial iteration decoding, and the bit error rate (BER) decreases as the number of cycles increases under a certain signal-to-noise ratio (SNR). However, after a certain number of cycles, the decoding performance will no longer improve, and redundant iterations significantly increase the decoding delay[14]. In LTE systems, turbo code adopts the idea of block parallel decoding and realizes partial parallel decoding of an entire code block through quadratic permutation polynomial interleaver[15]. However, the decoding algorithm's inherent serialization significantly limits throughput. The full parallel decoding algorithm of turbo codes, proposed in reference[16], divides the size of each subblock of the entire code block into 1 bit, achieving the highest eight-degree parallelism. However, the algorithm is implemented at the expense of performance, and parallelism has an upper limit. Ngo, Li et al[17,18]. conducted in-depth research and improved full parallel decoding algorithms, including the improved EXIT graph and turbo equalization algorithm. However, to achieve the performance of the traditional serial decoding algorithm, a full parallel decoding algorithm must increase the number of decoding iterations by approximately six times[19].

Although the channel condition is complex, deep neural networks are a general function approximator with excellent algorithm learning ability[20,21]. O 'Shea et al[22]. proposed a new idea for designing a communication system, namely, designing a physical layer through deep learning (DL). Fei Liang et al[23] designed an iterative belief propagation-convolutional neural network architecture using convolutional neural networks (CNNs) to decode low-density parity-check codes in correlated noise. Because of DL technology's continuous evolution, channel decoding based on DL has shown competitive performance[24]. Xiang Zhang et al[25] used a CNN to decode turbo codes in full parallel instead of traditional state transition. When the training set includes all datasets, the BER performance of their method is far less than that of one iteration BCJR algorithm.

To solve the above problems, this paper proposes a turbo code decoding algorithm based on the LSTM network, which transforms the decoding process into a big data classification and recognition problem. A neural network replaces the traditional decoding without prior coding knowledge, and the decoding algorithm model is obtained by directly using the received data and original bits of the receiver. The model has better BER performance and lower complexity. In addition, the communication system for different types of turbo codes has some generalization ability, which greatly improves the decoding efficiency.

2. Turbo Decoding Problem Description

Fig.1 shows the structure of the turbo code encoder in the 3GPP protocol. The encoding structures of the two-component turbo codes are identical, and they are cyclic system convolutional codes (2, 1, 3) with a code rate of 1/2.

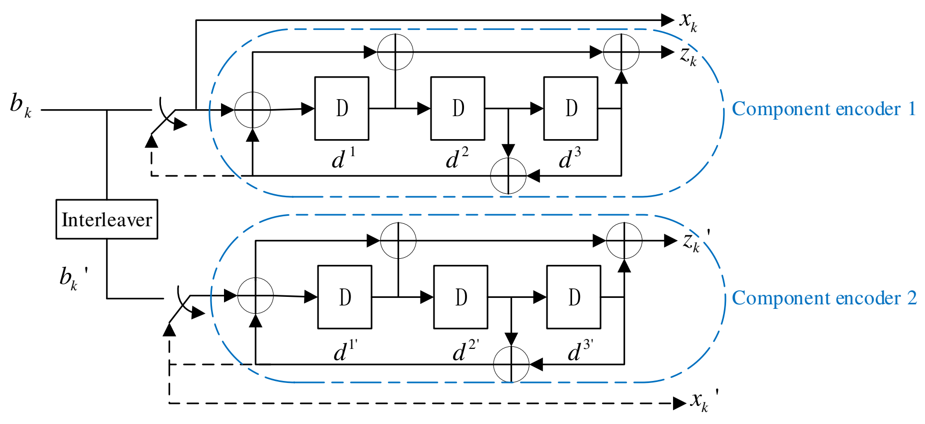

Fig.1 *Encoder structure of standard turbo code*

According to the coding structure in Fig.1, the following relationship can be deduced:

$$\begin{cases} z_k = d_k^1 + d_k^2 + b_k \\ d_{k+1}^1 = b_k + d_k^2 + d_k^3 \\ d_{k+1}^2 = d_k^1 \\ d_{k+1}^3 = d_k^2 \end{cases} \tag{1}$$

Where d_{k+1}^1 is the state of the first register at a time k. On the basis of Equation (1.1), we can deduce the following equation:

$$z_{k+1} = d_{k+1}^1 + d_{k+1}^2 + b_{k+1} = b_k + b_{k+1} + d_k^1 + d_k^2 + d_k^3 \tag{2}$$

In the same way,

$$\begin{cases} z_{k+2} = b_{k+1} + b_{k+2} + d_k^1 + d_{k+1}^1 + d_k^3 \\ z_{k+3} = b_{k+1} + b_{k+2} + b_{k+3} + d_k^1 + d_{k+1}^1 + 2d_k^3 \end{cases} \tag{3}$$

The following results can be obtained by combining Equations (1), (2), and (3),

$$\begin{cases} b_k = z_k + d_k^1 + d_k^2 \\ b_{k+1} = b_k + z_{k+1} + d_k^1 + d_k^2 + d_k^3 \\ b_{k+2} = b_k + b_{k+1} + z_{k+2} + d_{k+1}^1 + d_k^3 \\ b_{k+3} = b_k + b_{k+1} + b_{k+2} + z_{k+3} + d_k^3 \end{cases} \tag{4}$$

When the register states are all unknown, when the encoder state transition is not recorded, the more known information there is, the less unknown information is needed to predict the next information bit. The traditional BCJR algorithm's decoding steps are serial, where only the branch transition probability calculation does not rely on the leading data to achieve full parallel operation. The forward and backward recursive factors must wait for the pre-bit and post-bit calculations to complete before performing the parallel operation, which increases the decoding delay.

Theoretically, to simplify the model and reduce its computational complexity, the noise in the field of a wireless communication system is assumed to be Gaussian noise. However, because the information is subject to all types of interference when transmitted via radio, the statistical characteristics of noise do not always follow a Gaussian distribution. Suppose the model is used to describe the actual noise. In that case, the extracted information will be distorted to varying degrees, degrading or even damaging the performance of the corresponding information processing algorithm. Therefore, it is critical to investigate the channel decoding algorithm in non-Gaussian noise.

In this paper, a neural network is introduced into the receiver of a communication system. The LSTM network structure with a feedback loop is used to decode the turbo code, allowing the value of the research variable to be directly learned from the training data rather than deduced through serial recursion. Finally, the proposed method achieves a lower decoding delay and higher decoding efficiency than the traditional serial decoding algorithm.

3. Design of Turbo Decoder based on LSTM Neural Network

When convolutional codes are used as component codes, turbo codes can achieve excellent performance. Convolutional codes are an important basis for studying turbo codes. According to the coding structure of turbo codes and the concept of component decoding in the MAP algorithm, two sub-decoders are designed in a series, a Neural network decoder for convolutional codes.

3.1 System model

This section presents the design of a neural network-based decoder based on component decoding. It stacks the LSTM decoder of multiple circulatory convolutional codes and finally realizes the iterative decoding of turbo codes.

An LSTM network can store the received signal characteristics at each time and the signal at the previous time when inputting the codeword in a time sequence. Therefore, each component decoder uses an LSTM network as the basic network. Suppose a single LSTM network can memorize the characteristics of signals received from the forward direction. In that case, a bidirectional LSTM structure can take signals received from the forward and reverse directions as input and store the characteristics of the received signals at each time, the previous time, and the next time. Fig.2 is a block diagram of the system model of the LSTM decoder of cyclic system convolutional code.

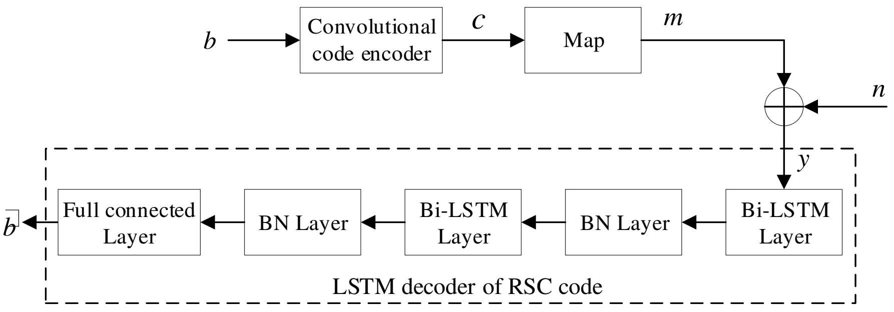

Fig.2 *System model of LSTM decoder for circulatory system convolutional codes*

The LSTM decoder connects a two-layer bidirectional LSTM network with a one-layer fully connected network and directly processes the transmitted information to obtain decoding results. The LSTM network can effectively reduce noise interference to the system, and then the fully connected network calculates the final decoding result from the information processed by the LSTM network. The decoder's performance mainly depends on the denoizing effect of the LSTM network on the received information and the feature extraction of information bits. Fig.3 shows the internal structure of the turbo decoder based on the LSTM network.

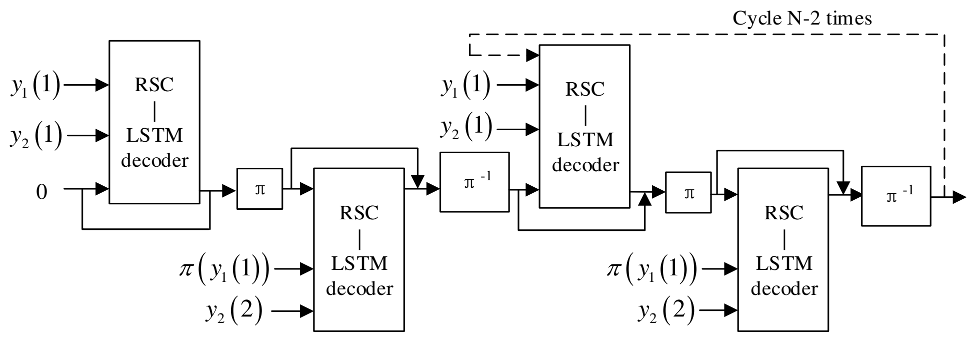

Fig.3 *Internal structure of turbo decoder based on an LSTM network*

In the first component decoding, for all $k \in [K]$, the LSTM decoding network uses the unified prior information b_k to estimate the posterior probability $\Pr(b_k \mid y_1(1), y_2(1))$. Then, in the second component decoding, the interleaved sequence $\pi(y_1(1))$ is used to estimate $\Pr(b_k \mid \pi(y_1(1)), y_2(2))$. At this time, the soft output of the first decoding should be used as a priori information. Repeat the above process, constantly improve the prediction output of codeword b_k until convergence, and finally estimate each bit to solve the BER at an SNR point. The BER at this point is taken as one of the performance criteria of the decoder.

3.2 Dataset generation

In a DL-based communication system, data at the transmitter's end are insufficient. However, at the receiver's end, due to the interference from channel noise, obtainable data become infinite, meeting the large data demand DL. The dataset in this study includes a training set and test set, and they are independent of each other. Before channel coding, a group of codeword sequences Y_i is encoded by the RSC polynomial function of each component, followed by bipolar mapping, and finally, through channel transmission to produce a group of noisy codeword sequences X_i. (X_i, Y_i) is a group of labeled training data, where X_i is the input data of the decoding network, and Y_i is the label data. The test set is obtained via conventional communication, and binary data are generated randomly and passed through a fixed SNR channel. Multiple SNR groups correspond to multiple test data groups. The LSTM network is trained using a large amount of labeled data; thus, the decoder can achieve the expected results.

3.3 Construction of decoding network

According to the design concept in 3.1, considering the complexity of the model building process, this section uses some high-level application programming interfaces to facilitate the rapid construction of the decoding network. Fig.4 shows the decoding network. The decoder structure of the two RSC codes is the same, except for the last output layer. The red circle represents the first RSC—the LSTM decoding network in Fig.3, and the green circle represents the second RSC—the LSTM decoding network. The output of the first RSC code decoder should be fed into the interleaver to randomly separate the burst errors while making the output of the two decoders independent of each other. Therefore, the design of the last layer of the two decoders is not identical. When training is completed, the *model.get_layer* command checks and returns all weight parameters of the two decoding networks. The *set_weights* command takes these weights as initialization parameters of the turbo decoding networks and loads them into each layer. The blue circle represents the interleaver, and the purple circle represents the de interleaver. The iterative process is implemented through a custom neural network Lambda layer implementation.

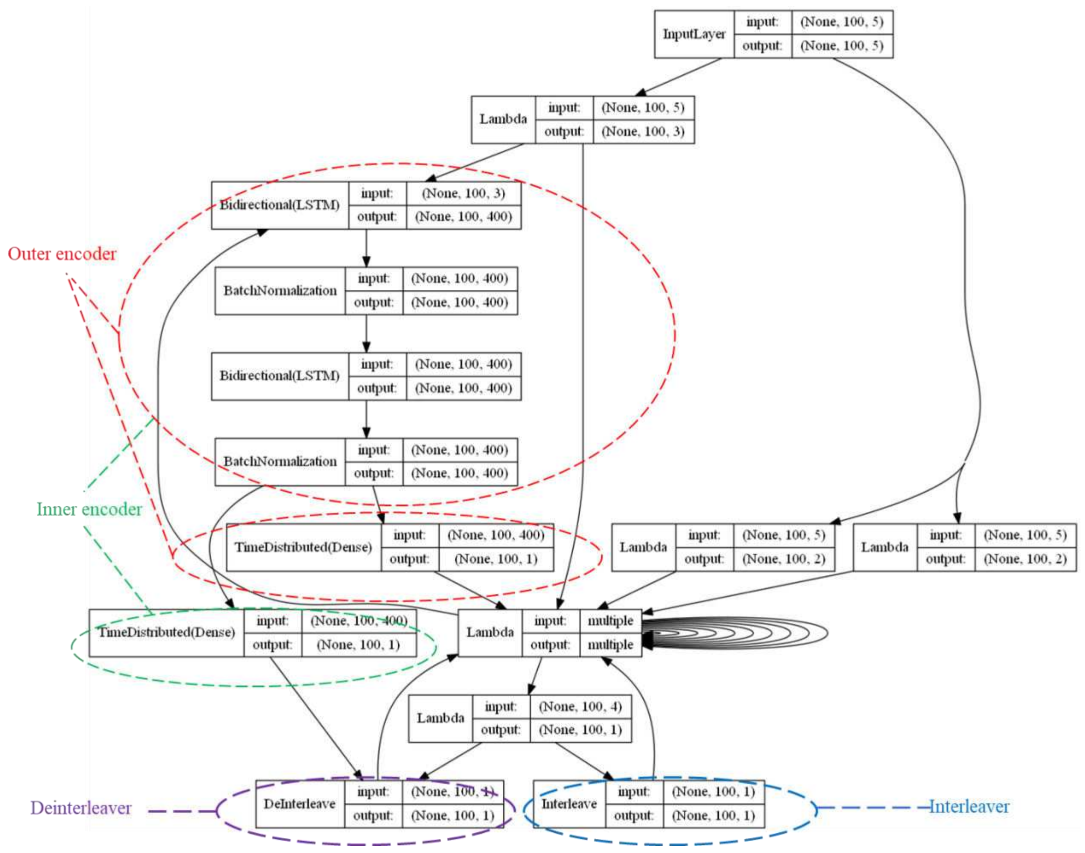
InputLayer | input: (None, 100, 5) | output: (None, 100, 5)
Lambda | input: (None, 100, 5) | output: (None, 100, 3)
Bidirectional(LSTM) | input: (None, 100, 3) | output: (None, 100, 400)
BatchNormalization | input: (None, 100, 400) | output: (None, 100, 400)
Bidirectional(LSTM) | input: (None, 100, 400) | output: (None, 100, 400)
BatchNormalization | input: (None, 100, 400) | output: (None, 100, 400)
TimeDistributed(Dense) | input: (None, 100, 400) | output: (None, 100, 1)
Lambda | input: (None, 100, 5) | output: (None, 100, 2)
Lambda | input: (None, 100, 5) | output: (None, 100, 2)
TimeDistributed(Dense) | input: (None, 100, 400) | output: (None, 100, 1)
Lambda | input: multiple | output: multiple
Lambda | input: (None, 100, 4) | output: (None, 100, 1)
DeInterleave | input: (None, 100, 1) | output: (None, 100, 1)
Interleave | input: (None, 100, 1) | output: (None, 100, 1)
Outer encoder
Inner encoder
Deinterleaver
Interleaver

Fig.4 *turbo code decoding network structure*

The decoding network in Fig.4 mainly includes three types of network structures: LSTM layer, full connection dense layer, and user-defined Lambda layer.

(1) LSTM layer, each LSTM layer adopts a bidirectional LSTM structure, enabling it to calculate the codeword sequence forward and backward at the same time. Under the premise of doubling the computational complexity, it can extract the related features of different information bits more accurately. When the LSTM layer inputs codeword in time sequence, it processes a three-dimensional tensor with three dimensions. The first dimension represents the number of samples to be processed, and its size is unknown when establishing the network, so it is none. The second dimension represents the number of time sequences, known as the number of frames to be processed by the loop layer, and its size is generally determined according to the time axis. The RSC component decoder decodes a fixed-length code block separately. In this section, the length of the code block is set to 100. Therefore, the second dimension size should be equal to the length of the code block and set to 100. The third dimension represents the characteristic number of each frame. The main research object of this paper is (7, 5, 7) turbo code, so the feature number of each frame is 5. In addition, a batch normalization layer is configured for each LSTM layer to accelerate network convergence and achieve a better decoding effect.

(2) Full connection dense layer calculates the extracted sequence-related information, selects sigmoid as the activation function, normalizes the output value of the network to between [0, 1], and estimates the information bits directly according to the output value, without the need for posterior probability.

(3) The user-defined Lambda layer of the neural network has no trainable parameters, only

realizes the function of adding noise or calculation, and transmits the results to the corresponding nodes in the next network layer. This helps simulate the iterative process.

3.4 Training of decoding network

The training method and training times of the decoding networks will affect the convergence speed and the optimal solution of network parameters, thereby affecting the performance of the decoding networks. Therefore, appropriate training methods should be used to determine the optimal solution of network parameters.

In this paper, stochastic gradient descent is used to train the networks. Each batch size contains 200 input units, and the networks update parameters after each batch size is processed. The network parameters are adjusted by the Adam optimization method. The initial learning rate is set to 0.001, and the learning rate is reduced to 1/10. For 10 to 15 epochs, the learning rate is reduced to 0.0001 every 5 epochs. When the epoch is more than 25, the learning rate decreases to 0.0000001. In addition, the training is terminated early if the verification loss does not decrease within 10 epochs to prevent overfitting. The binary cross-entropy is the loss function, and the training process goal is to reduce the loss as much as possible. The activation function of the LSTM layer is tanh with a faster convergence speed. Sigmoid is selected as the activation function of the dense layer, so the output value of the networks is between [0–1].

The RSC component decoding network decodes a code block separately. In this study, the length of the code block is set as 100 or 1000. Then, the BER performance analysis of the LSTM decoder is also based on the corresponding length of the code block.

4. Performance simulation and result analysis

Based on the structure and parameter settings of the decoding networks in the above section, the LSTM decoder is simulated and implemented for different turbo codes in the Gaussian white noise and distributed noise environment , respectively. To compare the decoder’s BER performance, the decoding results of the traditional BCJR algorithm under the same conditions are provided. The LSTM decoder’s performance, in terms of BER performance and computational complexity, will be analyzed in this study. The entire simulation process is divided into five parts: data preprocessing, network training, network testing, hard decision, and BER calculation. Fig.5 shows the turbo code decoding process based on LSTM.

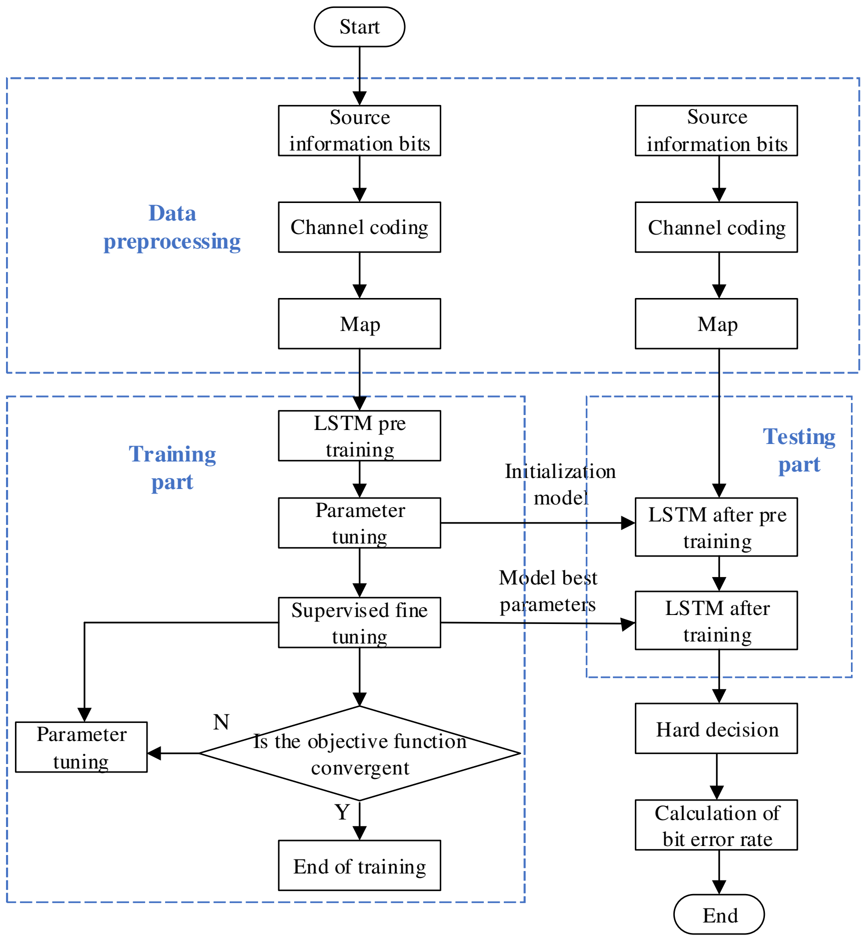

Fig.5 *turbo decoding process based on LSTM network*

4.1 Performance of LSTM decoder in Gaussian white noise channel

In the simulation experiment, taking the turbo code in this section (7, 5, 7) as an example, data of 10^4 sizes are selected to generate training data that meet the requirements. The training SNR is set to -1 dB, and the length of the code block is set to 100. Once obtained, the best network parameters are loaded into the turbo code decoding network as initial weights. Then, 100 turbo codes with a block length of 100 are used for end-to-end training, and the number of iterative training epochs is 30. The change of training accuracy and loss value with the number of iterations can be observed in real-time through the *hist* command. Other parameter settings are consistent with the above. Then, the test set is sent into the LSTM decoder, and the BER of each SNR point is calculated. The range of SNR is -1.5–2 dB, and each SNR point corresponds to data with a 10^5 size. Also, 100 turbo codes with a block length of 1000 are selected for simulation to analyze the applicability of the LSTM decoder.

When the block length is 100, the BER curve of (7, 5, 7) turbo code using the LSTM decoder is as shown in Fig.6. The neural network-based decoder proposed by Xiang Zhang is called a CNN decoder. In addition, the decoding results of the traditional BCJR algorithm under the same conditions help compare the neural network-based decoder's performance. Fig.7 shows the BER curves of the two decoding methods when the code block length is 1000.

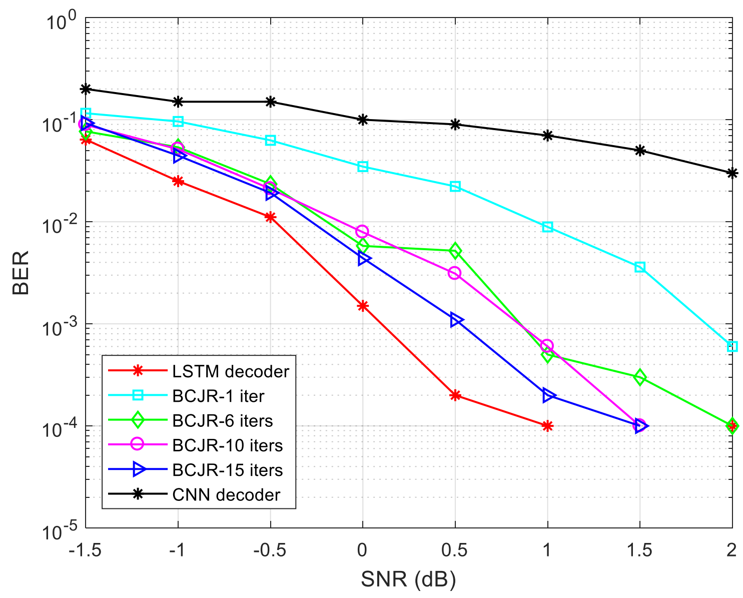

Fig.6 *Comparison of bit error rates of three decoding methods when the code block length is 100*

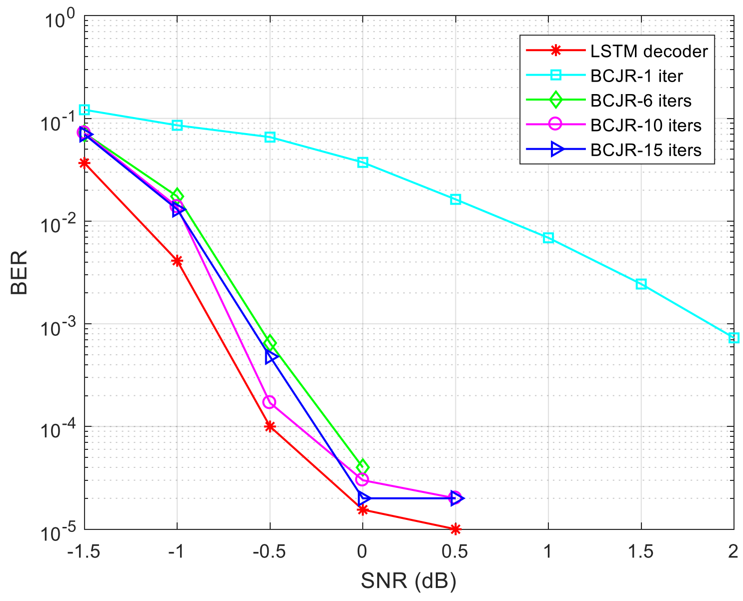

Fig.7 *Comparison of bit error rates between LSTM decoder and BCJR when block length is 1000*

Fig.6 shows the curve of the BER of the LSTM decoder and BCJR algorithm for 1, 6, 10, and 15 iterations and the BER of the CNN decoder with the SNR in the case of the Gaussian white noise channel, where the code block length is 100, the horizontal SNR (dB) represents the SNR. The vertical coordinate BER represents the value of BER. As shown in Fig.6, the BER of the LSTM decoder in the SNR range is lower than those of the other two decoding methods. When the BER is the same, the LSTM decoder has a 0.5 dB performance improvement over the BCJR decoding algorithm after 15 iterations at BER = 10^{-4}. It achieves 1 dB performance improvement over the BCJR decoding algorithm after 6 iterations. When the SNR is 0.5 dB, the BER of the LSTM decoder is 3 orders of magnitude lower than that of the CNN decoder.

Fig.7 shows the BER versus SNR curves of the LSTM decoder and BCJR algorithm for 1, 6, 10, and 15 iterations , respectively, in the same channel environment and with a block length of 1000. Fig.7 shows that the LSTM decoder achieves better BER performance when the code block length is 1000. When the SNR is -0.5 dB, the BER is 2 orders of magnitude lower than that when the code block length is 100 at the same SNR, which is sufficient to prove the excellent performance of turbo code in a low SNR environment. Furthermore, the LSTM decoder outperforms the BCJR decoding algorithm with 15 iterations in the entire SNR range. The SNR performance of the LSTM decoder is 0.4 dB higher than that of the BCJR decoding algorithm with 6 iterations at BER = 10^{-4}.

Decoding performance depends on the result of the BER and decoding costs, such as time complexity and hardware equipment costs. This section compares the different methods' decoding efficiency in terms of computational complexity. The calculation complexity index is set to the GPU time required by the above two decoding methods in the Gaussian white noise channel. Table 1 shows the various decoding methods' calculation complexity.

Table 1 *Operational complexity of different decoding modes*

Index / Decoding mode	GPU Time (Second)	
	Block Length = 100	Block Length = 1000
LSTM Decoder	13.8636	154.0169
BCJR-1 iter	1.7495	16.7617
BCJR-6 iters	9.3572	92.8210
BCJR-10 iters	15.3860	152.3455
BCJR-15 iters	23.2287	228.0381
BCJR-20 iters	30.5942	302.9107
BCJR-30 iters	45.3326	456.2875

Table 1 shows the decoding time required for the LSTM decoder and BCJR algorithm for 1, 6, 10, 15, 20, and 30 iterations, with block lengths of 100 and 1000, respectively. Table 1 shows that the longer the code block length, the longer the decoding time required by the two decoding methods. This is because the amount of computation required for decoding increases as the code block length increases. Comparing the decoding time of the same block length, the LSTM decoder requires more GPU time than the BCJR algorithm, with less than 10 iterations, and less GPU time than the BCJR algorithm, with 10, 15, 20, and more iterations. Combined with the above analysis, the BER performance of the LSTM decoder is always better than the BCJR algorithm in the same conditions. Although the former's decoding time is 10 times that of the latter, its BER is significantly better. This proves that the neural network-based decoding method has lower computational complexity and higher decoding efficiency than the traditional decoding method.

In conclusion, the decoding performance of the decoder based on the LSTM network is better than that of the traditional decoding method in the Gaussian white noise channel.

4.2 Bit error rate performance of LSTM decoder under t-distributed noise channel

This study mainly investigates a type of non-Gaussian noise that obeys the student distribution. t-distribution is a group of curves, and its shape change is related to the degree of freedom v. The smaller the degree of freedom, the lower the distribution curve; the larger the degree of freedom, the closer the distribution curve is to the standard normal distribution curve. Therefore, this section combines this property to determine whether the LSTM decoder's BER performance is affected by the degree of freedom to verify its robustness.

In this section, the turbo codes (7, 5, 7) are simulated for 3 and 5 degrees of freedom, respectively. The structure of the LSTM decoder is identical to that shown in Fig.4. When the degree of freedom $v = 3$, 10^4 groups of noisy codewords are collected as training data in the simulation environment with an SNR of 8 dB. The output data are generated in the same way, as described in Section 3.1. The data transmitted by the t-distribution noise channel are selected as the test set, and 10^5 sets of test data are collected at each SNR point. Similarly, we analyzed the BER performance of the LSTM decoder and BCJR algorithm when the block length is 100 and 1000, respectively. Other parameters are set per the previous section. When the block length is 100, the BER performance curve of the LSTM decoder and BCJR algorithm is shown in Fig.8. When the code block length is 1000, the BER curve of the two decoding methods is shown in Fig.9.

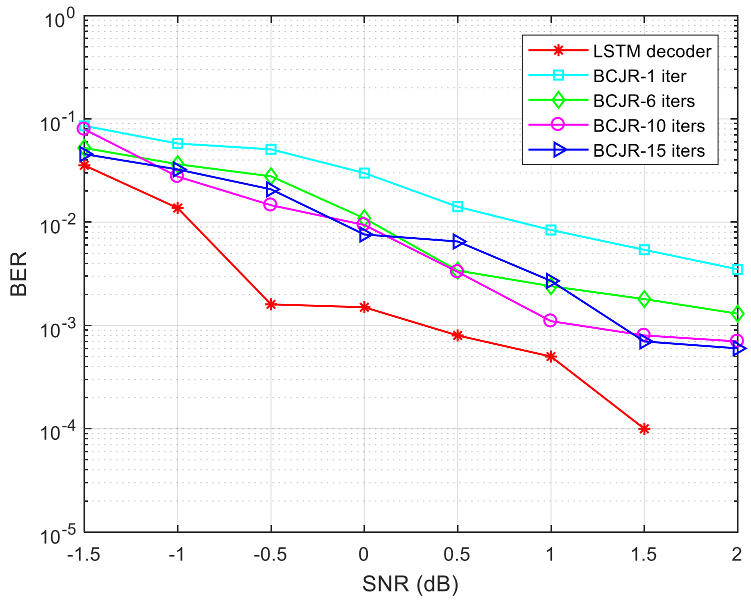

Fig.8 *Comparison of bit error rates between LSTM decoder and BCJR when v is 3 and code block length is 100*

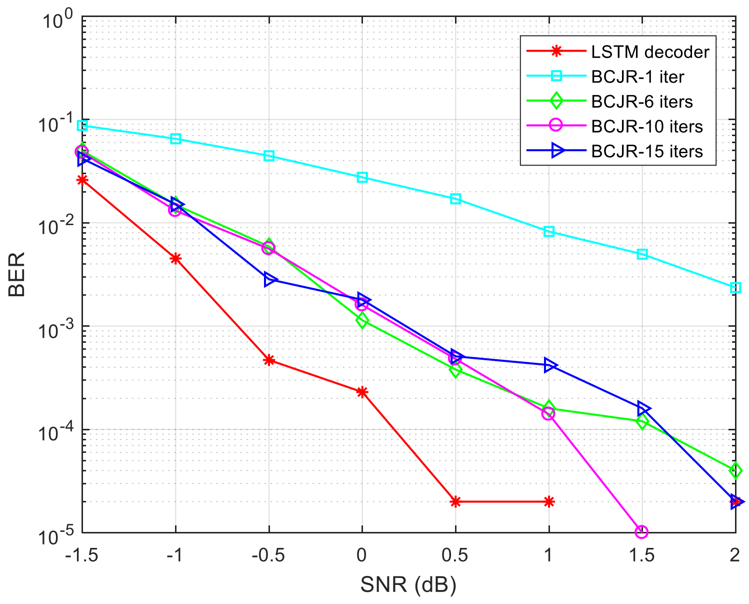

Fig.9 *Comparison of bit error rates between LSTM decoder and BCJR when v is 3 and code block length is 1000*

Fig.8 shows the BER versus SNR curve for the LSTM decoder and BCJR algorithm in 1, 6, 10, and 15 iterations, respectively, in the t-distribution noise channel and a code block length of 100. The analysis shows that the BER performance of the LSTM decoder is better than that of the BCJR algorithm with 15 iterations, and the performance of the traditional decoding algorithm is degraded in this non-Gaussian noise environment. Compared with the decoder's BER (Fig.6), when the BER drops to 10^{-4} order of magnitude, the SNR is 1 dB. In Fig.8, the required SNR is 1.5 dB, indicating that the LSTM decoder performance is also degraded in the non-Gaussian noise environment; however, it is still better than the traditional decoding algorithm in the same conditions.

Fig.9 compares the BER results of the two decoding methods when the code block length is 1000. The analysis shows that the performance of the LSTM decoder at BER = 10^{-4} is 1.5 dB higher than that of the BCJR decoding algorithm with 15 iterations and is much better than that of the BCJR decoding algorithm with 1 iteration.

When $v = 5$, the generation of training and test data in the simulation experiment is consistent with the above, and the training SNR is 8 dB. The BER performance comparison between the LSTM decoder and BCJR algorithm is shown in Fig.10 and Fig.11.

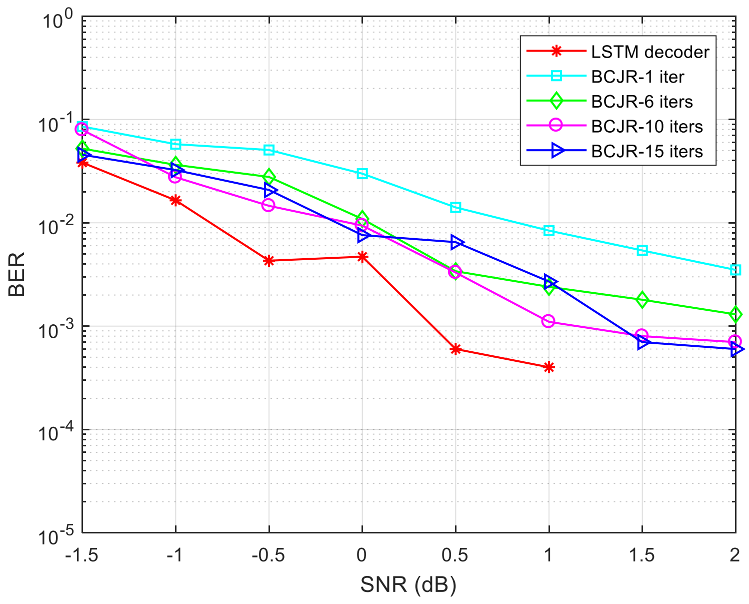

Fig.10 *Comparison of bit error rates between LSTM decoder and BCJR when v is 5 and code block length is 100*

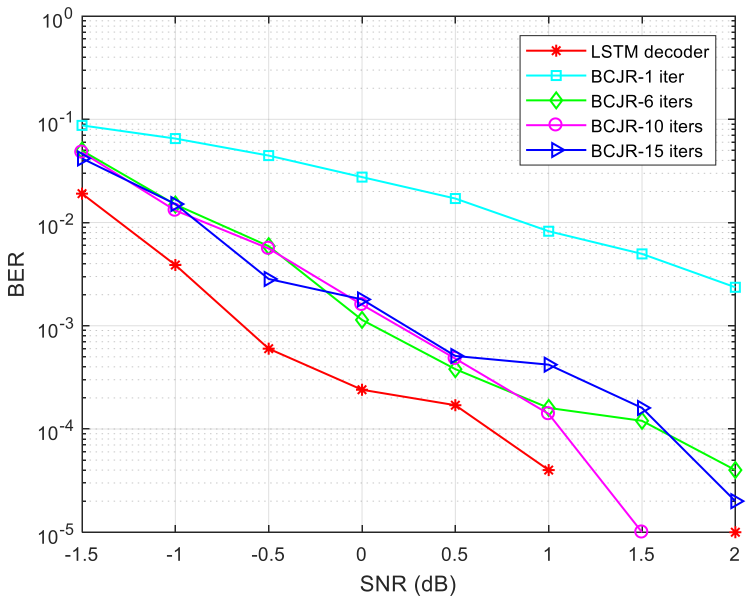

Fig.11 *Comparison of bit error rates between LSTM decoder and BCJR when v is 5 and code block length is 1000*

Fig.10 and Fig.11 show the BER versus SNR curves for the above two decoding methods when the code block length is 100 and 1000, respectively. Fig.10 shows that the BER performance of the LSTM decoder is better than that of the BCJR algorithm with 15 iterations. In addition, compared with the results in Fig.8, the performance of the LSTM decoder is degraded, which indicates that the change of degree of freedom affects decoding performance. An increase in the degree of freedom degrades the performance of the two decoding methods. In Fig.11, the LSTM decoder's performance at BER = 10^{-4} is 1 dB higher than the BCJR algorithm with 15 iterations. The performance is much better than that of the BCJR algorithm with 1 iteration.

In conclusion, the BER performance of the LSTM decoder and traditional BCJR algorithm has different degrees of degradation in the t-distribution noise channel. Furthermore, the performance of the LSTM decoder is still better than that of the BCJR algorithm. When the BER is the same, the performance of the LSTM decoder is better than that of the traditional BCJR decoding algorithm, and it is better than that of the traditional algorithm in the Gaussian white noise channel. In addition, the performance of the LSTM decoder is affected by the degree of freedom. The smaller the degree of freedom, the greater the improvement of BER performance. The larger the degree of freedom, the smaller the improvement of BER performance. The above results show that the LSTM decoder is robust to some extent.

4.3 Bit error rate performance of LSTM decoder under different turbo codes

This section studies the BER performance of turbo codes with different length constraints. It simulates the transmission process of (7, 5, 7), (7, 5, 6), (7, 5, 5), (7, 5, 4), and (7, 5, 3) turbo codes in the Gaussian white noise channel to explore the applicability and robustness of the proposed decoder. These codes have the same bit rates and different memory depths. The network model and parameter settings are consistent with those in Section 3.4. The BER curves of the above five turbo codes are obtained through simulation when they are transmitted in the Gaussian white noise channel, as shown in Fig.12.

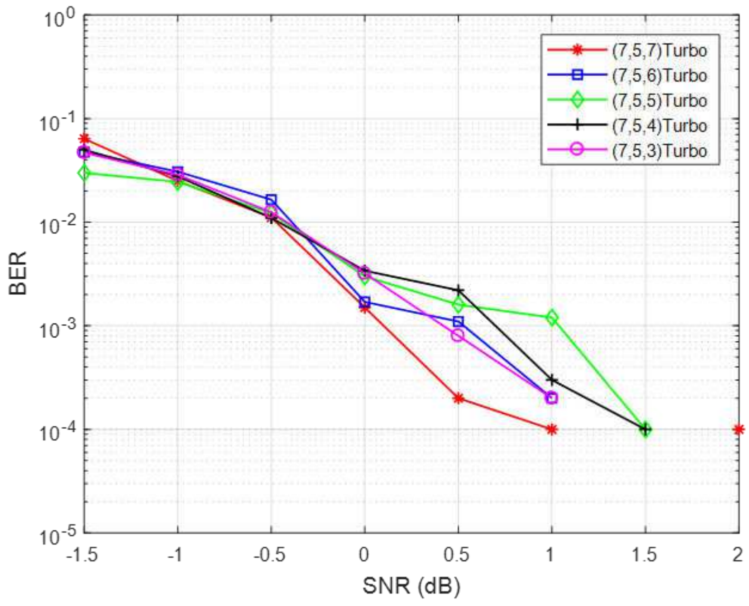

Fig.12 *Comparison of bit error rates of different turbo codes when the code block length is 100*

Fig.12 shows the BER curves of the LSTM decoder with varying SNRs when turbo codes with different constraint lengths are transmitted in the Gaussian white noise channel. Fig.12 shows that the LSTM decoder can achieve good decoding performance on the turbo codes with a constraint length of 3. The BER of the decoder shows no obvious downward trend when the constraint length is reduced from 7 to 3. However, the traditional decoding algorithm could not effectively decode the received signals with varying SNRs when the constraint length is 3. The shorter the constraint length, the worse the decoding performance of the traditional decoding algorithm. Therefore, the LSTM decoder has certain applicability for a variety of turbo codes.

5. Conclusion

In this study, a turbo decoder based on an LSTM network is proposed to solve the high decoding delay problem caused by the poor parallelism of existing decoding algorithms. From the viewpoint of component decoding, this decoder stacks multiple LSTM decoders of cyclic system convolutional codes and eventually realizes the iterative decoding of turbo codes. The simulation results show that this neural network-based decoding method has lower BER and computational complexity than existing decoding methods in the same conditions. Furthermore, the LSTM decoder decodes the received signal directly rather than processing the related noise alone, which solves the problem of performance degradation of the traditional decoding algorithm in the non-Gaussian noise environment. The online network only needs to update the offline network. After training the network parameters, the signals in different noise environments can be switched, which greatly improves the decoding efficiency.

Acknowledgments

We gratefully acknowledge the anonymous reviewers who read the drafts and provided many helpful suggestions.

Declarations

Funding:

This work is supported by the Natural Science Foundation of Shanghai (19ZR1454000).

Conflicts of interest:

The authors have no relevant financial or non-financial interests to disclose.

Availability of data and material:

The datasets generated or analyzed and material during this current study are available from the corresponding author on reasonable request.

Code availability:

The codes during this current study are available from the corresponding author on reasonable request.

Authors' contributions:

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Li Zhang, Weihong Fu, Fan Shi, Chunhua Zhou, and Yongyuan Liu. The first draft of the manuscript was written by Li Zhang and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

[1] WEBER S, ANDREWS J G. Transmission capacity of wireless networks. Foundations & Trends® in Networking, 2012, 5(2-3): 3593-3604.

[2] SAMPEI S, HARADA H. System design issues and performance evaluations for adaptive modulation in new wireless access systems. Proceedings of the IEEE, 2007, 95(12): 2456-2471.

[3] SHAFI M, MOLISCH A.F, and SMITH P.J. 5G: A tutorial overview of standards, trials, challenges, deployment, and practice. IEEE journal on selected areas in communications. 2017, 35(6): 1201-1221.

[4] WANG T, WEN C.K, and WANG H. Deep learning for wireless physical layer: Opportunities and challenges. China Communications, 2017, 14(11): 92-111.

[5] WU Y. Implementation of parallel and serial concatenated convolutional codes. Virginia Polytechnic Institute and State University, 2000.

[6] BERROU C, GLAVIEUX A, and THITIMAJSHIMA P. Near shannon limit error-correcting coding and decoding: Turbo-codes.1. Proceedings of ICC'93-IEEE International Conference on Communications, 1993: 1064-1070.

[7] ROBERTSON P. Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes. in 1994 IEEE GLOBECOM Communications: The Global Bridge, San Francisco. IEEE, 1994: 1298-1303.

[8] BERROU C, GLAVIEUX A. Near optimum error correcting coding and decoding: Turbo-codes. IEEE Transactions on communications,1996, 44(10): 1261-1271.

[9] MACKAY D.J, NERL R.M. Near Shannon limit performance of low density parity check codes. Electronics letters, 1996, 32(18): 1645-1646.

[10] HAGENAUER J, HOEHER P. A viterbi algorithm with soft-decision outputs and its applications. in 1989 IEEE Global Telecommunications Conference and Exhibition' Communications Technology for the 1990s and Beyond', Dallas. IEEE, 1989: 1680-1686.

[11] BAHL L.R, COCKE J, and JELINEK F. Optimal decoding of linear codes for minimizing symbol error rate (Corresp.). IEEE Transactions on Information Theory, 2003, 20(2): 284-287.

[12] ROBERTSON P, VILLEBRUN E, and HOEHER P. A comparison of optimal and sub-optimal MAP decoding algorithm operating in the Log domain. in Proceedings IEEE International Conference on Communications ICC '95, Seattle. 1993: 1009-1010.

[13] MACKAY D.J, NERL R.M. Near Shannon limit performance of low density parity check codes. Electronics letters, 1996, 32(18): 1645-1646.

[14] PROAKIS J.G. Digital communication. Electronic Industry Press, 2006.

[15] NIMBALLKER A, BLANKENSHIP Y and CLASSON B. ARP and QPP Interleavers for LTE Turbo Coding. in 2008 IEEE Wireless Communications & Networking Conference, Las Vegas. IEEE, 2008: 1032-1037.

[16] NGO H A, MAUNDER R G, and HANZO L. Extrinsic information transfer charts for characterizing the iterative decoding convergence of fully parallel Turbo decoders. IEEE Access, 2015, 3: 2100-2110.

[17] LI A, XIANG L, and CHEN T. VLSI implementation of fully parallel LTE Turbo decoders. IEEE Access,2016, 4: 323-346.

[18] WEITHOFFER S, NOUR C A, and WHEN N. 25 Years of turbo codes: from Mb/s to beyond 100 Gb/s. 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC), 2019: 1-6.

[19] RUSSAKOVSKY O, DENG J, and SU H. Imagenet large scale visual recognition challenge. International journal of computer vision, 2015, 115(3): 211-252.

[20] HE H, WEN C.K, and JIN S. Deep learning-based channel estimation for beamspace mmWave massive MIMO systems. IEEE Wireless Communications Letters, 2018, 7(5): 852-855.

[21] O'SHEA T, HOYDIS J. An introduction to deep learning for the physical layer. IEEE Transactions on Cognitive Communications and Networking, 2017,3(4): 563-575.

[22] LIANG F, SHEN C, WU F. An iterative BP-CNN architecture for channel decoding. IEEE Journal of Selected Topics in Signal Processing, 2018, 12(1): 144-159.

[23] GRUBER T, CAMMERER S, and HOYDIS J. On deep learning-based channel decoding. in 2017 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore. IEEE, 2017: 1-6.

[24] JEON Y.S, HONG S.N, and LEE N. Supervised-learning-aided communication framework for massive MIMO systems with low-resolution ADCs. IEEE Transactions on Vehicular Technology, 2018, 67(8): 1-1.

[25] ZHANG X, LUO T. A RNN Decoder for Channel Decoding under Correlated Noise. in 2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops), Changchun. 2019: 30-35.