IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 2 December 2022; revised 18 March 2023 and 14 May 2023; accepted 14 May 2023.
Date of publication 22 May 2023; date of current version 8 June 2023.

The associate editor coordinating the review of this article and approving it for publication was J. Zhang.

Digital Object Identifier 10.1109/TMLCN.2023.3279051

A Neural Network Empowered Belief Propagation Algorithm Optimized for Short-Cycles in Tanner Graph

HAO XU[1], YING LI[1], BIN TAN[2], JUN WU[3] (Senior Member, IEEE), AND DIE HU[3]

[1]College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China
[2]College of Electronics and Information Engineering, Jinggangshan University, Ji'an 343009, China
[3]School of Computer Science, Fudan University, Shanghai 200433, China

CORRESPONDING AUTHORS: D. Hu (hudie@fudan.edu.cn) AND J. Wu (wujun@fudan.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61831018, Grant 62261030, Grant 62271155, and Grant U21A20452; in part by the Shanghai Science and Technology Innovation Action Plan under Project 21220713100; and in part by the Natural Science Foundation of Jiangxi, China, under Grant 20212ACB212001.

ABSTRACT Short-cycles in Tanner graphs have a direct impact on the accuracy and effectiveness of the belief propagation (BP) algorithm, as they diverge the BP algorithm by disrupting the independency of message transmission. In this paper, we present a Neural Network Empowered BP (NNE-BP) algorithm as an alternative to the conventional BP algorithm, which can approach the optimum form of BP in short-cycles. The earlier deep learning (DL) based decoder treated the entire decoding process as a classification issue, making it effective only in short block lengths and with limited generalization. In the proposed NNE-BP, we design two neural networks, namely the horizontal network and vertical network, to learn, respectively, the optimized horizontal and vertical processes for single nodes in the presence of cycles. These two networks operate in an iterative fashion, allowing them to approach a theoretical message propagation equation without assuming independent message transmission. Compared to the classification based DL decoder, the proposed network topology is no longer bundled with any specific Tanner graph, and the number of network parameters are greatly reduced. The obtained experimental results show that the proposed NNE-BP outperforms the conventional BP decoder, especially in the case of short-cycles.

INDEX TERMS Neural networks, belief propagation, channel coding.

I. INTRODUCTION

LOW-DENSITY parity-check (LDPC) code is a linear block code with a sparse check matrix proposed by Gallager in 1963 [7]. It has been widely applied to communication systems [4] as its bit error rate (BER) performance is close to the Shannon limit and attains high throughput [20], [21], [22]. LDPC code is usually decoded with the BP or min-sum (MS) algorithm. The BP algorithm achieves excellent BER performance while performing a large amount of calculation [23]. The MS algorithm simplifies the horizontal process in the BP algorithm and significantly reduces the decoding complexity, but inevitably its performance is degraded. Later on, the MS algorithm was improved by incorporating a correction factor to make up the performance loss [26]. The efficiency and accuracy of the BP process is affected by the cycles in the Tanner graph. Since the BP process is based on the assumption of acyclic graphs, short-cycles cause rapid return of information, which destroys the independence of message transmission, thus affecting the convergence of the BP process. Most of the cycle elimination algorithms and matrix construction methods intend to eliminate the 4-cycles or 6-cycles [29], [30], which cannot completely eliminate the influence caused by the short-cycles of Tanner graph.

In recent years, the research on the LDPC code is not limited to the improvement of the conventional decoding algorithm [16], [17], [18], [19] only, but the deep learning has also been introduced in the field of channel decoding [8], [9], [10], [11]. At present, the DL-based decoding algorithms can be divided into three categories. The first category considers the entire decoding process as a single classification problem,

where a deep neural network (DNN) is used to fit the entire decoding process [2]. The main function of DNN is to map the log-likelihood ratio (LLR) of the entire code block onto the corresponding information bit stream. Such ''black box'' algorithms do not fully utilize the information obtained from the channel encoding, such as the Tanner graph. In the second category of the DL-based decoding algorithms, the Tanner graph is integrated into the structure of DNN. The common practice is to parameterize the decoding process of the conventional algorithm, or to include the learnable parameters in the traditional decoding algorithm. Nachmani et al. [5] constructed a parameterized BP decoder by manually building the connections among the neurons in DNN according to the target Tanner graph, and found that the BER performance is better than that of the BP algorithm in short-code with fewer iterations. The Recurrent Neural Network (RNN) was used to replace the DNN structure in a later work [3], which reduces the number of parameters, thus further reduces the decoding delay. Wang et al. [1] parameterized the MS decoding process by converting the originally fixed correction factor into learnable parameters, which resulted in a significant reduction in BER. Similar practices were applied to non-binary LDPC codes also. Liu and Chen [24] designed a neural network decoding architecture based on the Tanner graph [5]. Watanabe et al. [11] offered different weights to the edges of the Tanner graph during the log-likelyhood ratio (LLR) updating to reduce the number of iterations. The DL-based decoding algorithms of the third category do not use DL directly for decoding, instead they improve the performance by cascading a convolutional neural network in front of the BP decoder [28] or DL decoder [27] for denoising.

Even though the DL decoder outperformed the BP/MS method in terms of BER or decoding delay, at the same time it paid the price of loss of generalization also. In order to fit the decoding operation of the entire code block into a classification task, the DL decoder must be responsible for both updating and transmitting information at the same time. However, different LDPC codes have different Tanner graphs correlating to different pathways of information transmission. In real-world circumstances, the LDPC codes are designed according to different task requirements. The conventional BP algorithm can handle all types of Tanner graphs with slightly different parameters, while the DL decoder can be designed only for one specific Tanner graph, as the activation relationship among the neurons [3], [5], [10] or the number of neurons in the hidden layer [1], [3], [5] is closely coupled with the specific Tanner graph. Furthermore, the low decoding latency of the DL decoders does not imply fewer calculations or parameters, but is realized by the high-performance processing platforms, and the number of neurons in the hidden layer is frequently positively associated with the number of edges [3], [5] or nodes [1] in the Tanner graph. Not only the number of parameters will increase as the length of the code increases, most DL decoders now use the training technique described in [2], which treats the decoding problem as the classification of the entire code block. When the length of the code is K, the space size of the sample becomes 2^K, which implies that the task's scale grows exponentially as the length of the code grows.

In order to improve the performance of the BP algorithm on the Tanner graph with cycles, we present a novel neural network empowered decoder, named as NNE-BP. Unlike in the earlier DL decoders, we no longer use any neural network to fit the decoding process of the entire code block. Instead, we design a network of an appropriate scale to fit the two most basic operations in the BP algorithm, namely the update processes of single check node (CN) and variable node (VN), also known as the horizontal and vertical processes, respectively. These processes are related only to the degrees of CN and VN. So, the proposed network structure does not need to address the code length and Tanner graph. As in the BP algorithm, the information between nodes is conveyed directly through the Tanner graph. We simply need to replace the Tanner graph for decoding different LDPC codes by using the same trained model. Hence, the generalization ability of the suggested design is nearly the same as in the conventional BP method. Simultaneously, the network scale is related only to the complexity of the horizontal and vertical processes in the BP algorithm, which is related only with the degree of the LDPC codes and not related with the code length.

Our main contributions are summarized below:

- We propose a novel NN-empowered decoder, named as NNE-BP, which can approach the optimized version of the BP algorithm with short-cycles present in the Tanner graph. Along with the appropriately designed neural network model, the training data will drive NNE-BP to approach a better performance with short-cycles.
- Unlike the earlier DL decoding algorithms, NNE-BP is not coupled with any specific Tanner graph, which can use the same trained model to decode different LDPC codes without fine tuning.
- NNE-BP focuses only on the information update process of each single CN&VN. So, the network complexity and training cost do not grow with any increase in the length of the code.
- With a learnable padding strategy, we extend NNE-BP to irregular LDPC codes, and achieve better BER performance than that of the BP algorithm when the irregularity is small.

The rest of this paper is organized as follows: we give a brief introduction of the conventional BP algorithm in Section II. Then, the proposed NNE-BP decoder is introduced in detail in Section III. The experimental settings and simulation results are presented in Section IV. In section V, we conclude the paper and list the future work.

II. BELIEF PROPAGATION

The message-passing decoders, such as the BP decoder, of an LDPC code are constructed by the Tanner graph that is associated with the parity check matrix H of the code. A Tanner graph, which includes CNs, VNs, and edges E,

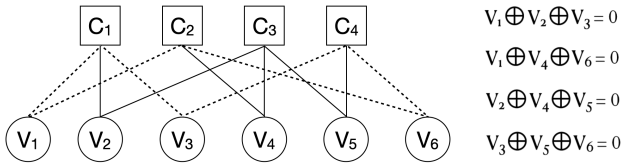


FIGURE 1. Tanner graph with 6-cycles.

is a graph representation of the parity check matrix. Besides, conventionally, the calligraphic font E is used to represent the set of edges. The edge $\mathrm{E}_{i,j}$ connects the ith CN and jth VN, corresponding to row i and column j of the H, that is $H_{i,j}$ = "1". During the decoding, the LLR information is updated iteratively between CN and VN through the following steps:

1) Calculate $LLR\left(VN_j\right)$ corresponding to the information bit VN_j according to the channel information Y_j as the initialization, $j = 1, 2, \ldots, N$, where N represents codeword length:

$$LLR\left(VN_j\right) = \log \frac{P\left(VN_j = 0|Y_j\right)}{P\left(VN_j = 1|Y_j\right)} \tag{1}$$

$$LLR\left(E_j\right) = LLR\left(VN_j\right) \tag{2}$$

where E_j represents all the edges connected to VN_j.

2) Horizontal process: calculate the CN-To-VN message as follows:

$$LLR\left(E_{i,j}\right) = 2\tanh^{-1}\left[\prod_{j' \in \mathrm{E}_{i/j}} \tanh\left(\frac{1}{2}LLR\left(E_{i,j'}\right)\right)\right] \tag{3}$$

where $\mathrm{E}_{i/j}$ represents all the edges connected to CN_i, excluding $E_{i,j}$.

3) Vertical process: calculate the VN-To-CN message as follows:

$$LLR\left(E_{i,j}\right) = LLR\left(VN_j\right) + \sum_{i' \in E_{j/i}} LLR\left(E_{i',j}\right) \tag{4}$$

where $E_{j/i}$ represents all the edges connected to VN_j, excluding $E_{i,j}$.

4) Message aggregation: calculate LLR for VN as follows:

$$LLR\left(VN_j\right) = LLR\left(VN_j\right) + \sum_{i \in \mathrm{E}_j} LLR\left(\mathrm{E}_{i,j}\right) \tag{5}$$

5) Parity check: determine whether the iteration is to be terminated. The estimated value of each VN can be calculated as follows:

$$\hat{VN}_j = \begin{cases} 1, & LLR\left(VN_j\right) \le 0 \\ 0, & LLR\left(VN_j\right) > 0 \end{cases} \tag{6}$$

$\hat{VN} = [\hat{VN}_1, \hat{VN}_2, \ldots, \hat{VN}_N]$, if $H \cdot \hat{VN}^T = 0$, T stands for matrix transpose, then the estimated value will be the output to complete the decoding. Otherwise, repeat Steps 2, 3 and 4 until the parity-check is passed or the maximum number of iterations is reached.

When there is no cycle in the Tanner graph, the BP algorithm can achieve the best decoding effect. However, the cycles are inevitable in practice, so that (3) and (4) no longer hold strictly. For example, in the Tanner graph as shown in **Fig. 1**, each CN represents a check equation assuming that each VN is independent of each other. Set $P(VN_i = 1) = p_i$ and $P(VN_i = 0) = (1 - p_i) = q_i$. Then, calculate the following:

$$P(VN_1 \oplus VN_2 \oplus VN_3 = 0) = q_1q_2q_3 + p_1p_2q_3 + p_1q_2p_3 + q_1p_2p_3 \tag{7}$$

It can be further simplified to:

$$P(V_1 \oplus V_2 \oplus V_3 = 0) = 0.5 + 0.5\prod_{i=1}^{3}(1 - 2p_i) \tag{8}$$

However, due to the existence of the cycles, it is difficult to maintain independency between VNs, and (7) can be rewritten as:

$$P(V_1 \oplus V_2 \oplus V_3 = 0) = \lambda_1 q_1q_2q_3 + \lambda_2 p_1p_2q_3 + \lambda_3 p_1q_2p_3 + \lambda_4 q_1p_2p_3 \tag{9}$$

where λ is a parameter introduced in the conditional probability as the correlation of VNs. For example, λ_1 can be expressed as:

$$\lambda_1 = \frac{P(V_1 = 0 \mid V_2 = 0, V_3 = 0)P(V_2 = 0 \mid V_3 = 0)}{P(V_1 = 0)P(V_2 = 0)} \tag{10}$$

Similarly, $\lambda_{2\sim4}$ can be calculated theoretically. Select appropriate parameter μ and further simplify (9) as:

$$P(V_1 \oplus V_2 \oplus V_3 = 0) = 0.5 + 0.5\prod_{i=1}^{3}\mu_i(1 - 2p_i) \tag{11}$$

Equations (3) and (4) are put into (8) without considering μ. Since it is difficult to calculate μ accurately in the Tanner graph with complex cycles, the conventional BP algorithm cannot achieve the theoretical performance in the actual Tanner graph.

Another drawback with the BP algorithm is the high complexity. To begin with, Equation (3) is a serial calculation process involving logarithmic and multiplication operations, which has a high degree of complexity. Furthermore, the BP algorithm necessitates numerous iterations when the channel conditions are bad.

III. PROPOSED NNE-BP DECODER

A. OVERVIEW

The BP algorithm is based on an acyclic graph, which is not an optimal information update process in the actual Tanner graph with cycles. The existing DL decoders have achieved excellent BER performance, while their generalization ability is poor for their coupling with a specific Tanner graph. At the same time, a large number of parameters are required when dealing with long codes. Inspired by the advantages of

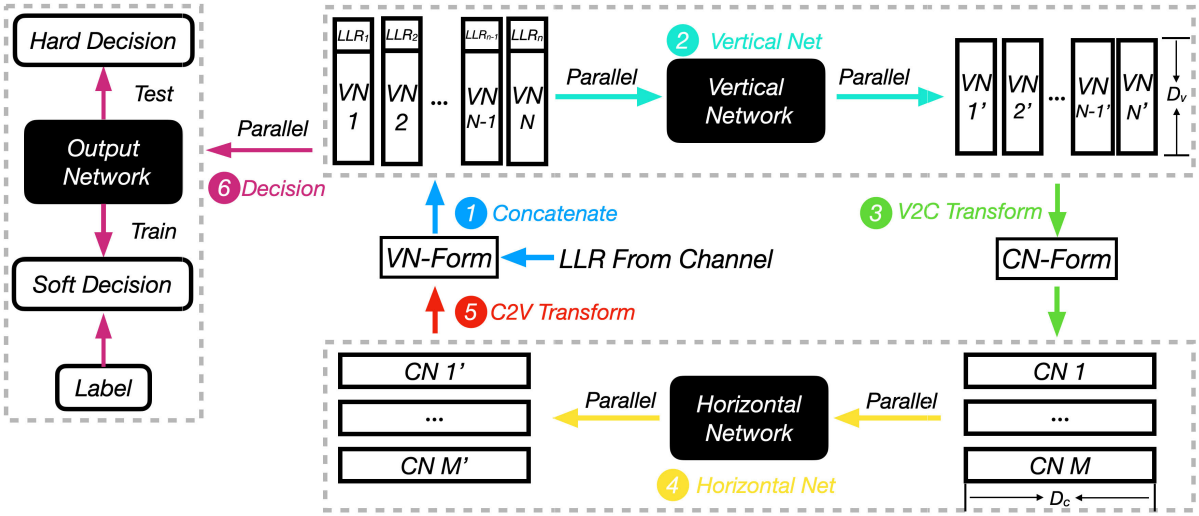


FIGURE 2. Workflow of the proposed NNE-BP.

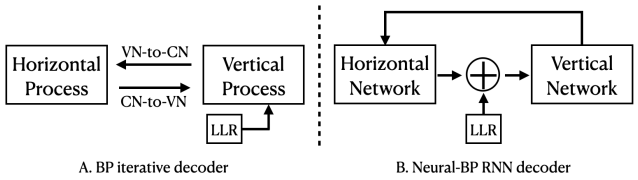


FIGURE 3. Structure of conventional BP and NNE-BP.

the above two types of methods, we propose the NNE-BP decoder as an optimized version of the BP algorithm. As illustrated in Fig. 3, the horizontal and vertical processes in the BP algorithm are replaced with appropriately designed neural networks, namely the horizontal network and vertical network, respectively. Those two networks are used to learn the weighted CN&VN updating mechanism in the practical Tanner graph, and the information is transmitted directly through the Tanner graph.

The proposed NNE-BP decoder adopts an end-to-end training strategy, including three sub-networks, namely the horizontal network, vertical network and output network. Consider them as three mapping functions: $f(x)$, $g(x)$, $z(x)$, respectively. Then:

$$E'_{i,:} = f(E_{i,:}) \tag{12}$$

$$E'_{:,j} = g(E_{:,j}, LLR_j) \tag{13}$$

$$P_j = \text{Softmax}(z(E_{:,j}, LLR_j)) \tag{14}$$

where $E_{i,:}$ represents all the edges connected to the ith CN, $E_{:,j}$ represents all the edges connected to the jth VN, $E'_{:,i}$/ $E'_{:,j}$ denote $E_{i,:}$/ $E_{:,j}$ updated by horizontal/vertical network. $P_j = [P_j^0, P_j^1]$ denotes the soft decision information of the jth VN, $f(x)$ and $g(x)$ are used to update LLR, and $z(x)$ is used for decision. We cannot train $f(x)$ and $g(x)$ alone as we do not know the exact information update mechanism in the real Tanner graph. However, the entire decoding process is differentiable. So, the binary cross entropy loss between P_j and real label y_j can be used to optimize the entire decoding process as:

$$\text{Loss} = -\frac{1}{N}\sum_j \left[y_j \times \log\left(P_j^1\right) + (1 - y_j) \times \log\left(P_j^0\right) \right] \tag{15}$$

So, the BP process in the real Tanner graph can be learned by minimizing the loss as given by (15). In the next section, we will elaborate the information update process and information transfer process of NNE-BP.

B. PROPOSED NNE-BP

Figure 2 shows the workflow of the proposed NNE-BP decoder, which contains the network structure and data transmission used in the decoding process. The network structure is represented by the black-filled box, whereas the data is represented by the non-filled box. Denote that the codeword length of the regular LDPC codes is N, the message word length is K, M = N − K. Then the parity check matrix can be expressed as $H[M, N]$ and its CN and VN degrees are D_c, D_v, respectively. Steps 1–6 below correspond, respectively, to processes 1–6 as shown in Fig. 2.

Step 0: Construct the VN-Form and CN-Form as shown in Fig. 4, where the ith column of the VN-Form stores the edge information of the ith VN, and the jth row of the CN-Form stores the edge information of the jth CN. For a deterministic Tanner graph, there is a unique mapping between the VN-Form and CN-Form. Initialize the VN-Form with zero. At the same time, LLR vector of size $[1, N]$ is obtained by using (1).

Step 1: Connect the LLR vector with the VN-Form through the column to obtain the input tensor for vertical network as shown in Fig. 5. Each column of the vertical input contains the LLR and edge information of a single VN, which is the

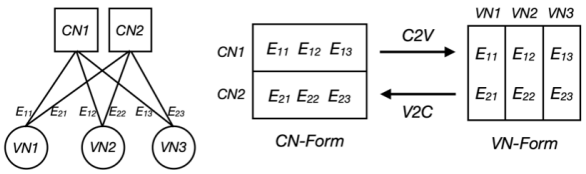


FIGURE 4. Construction of VN-form and CN-form.

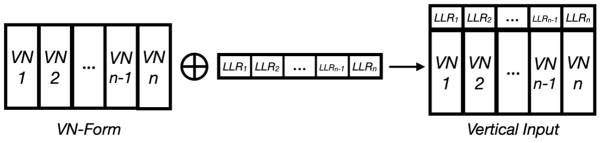


FIGURE 5. Tensor concatenation for vertical network.

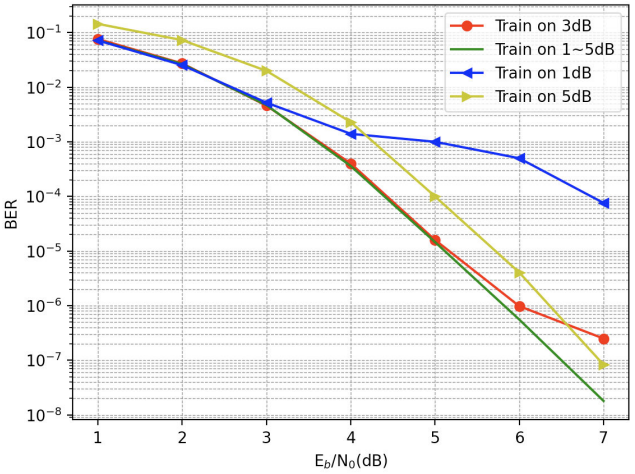


FIGURE 6. Influence of SNR range of training dataset on BER.

same with the information required by the BP algorithm to complete a vertical process (4).

Step 2: The vertical network plays a similar role to the vertical process in the BP algorithm, whose role is to update the edge information of each VN. The vertical input obtained in Step 1 is delivered by the columns to the vertical network, and the output is the updated edge information with the dimension of $[1, D_v]$. Note that (4) in the BP algorithm updates one edge at a time, whereas the vertical network calculates all the edges of each VN node at once. The reason for doing the vertical iteration first is that in Step 1 we performed only the zero initial to the VN-Form, and the vertical iteration can serve as an initializer for the VN-Form by using LLR given by (2).

Step 3: Reorganize the updated VN-Form to obtain the CN-Form according to the Tanner graph as show in Fig. 4. Each row of the CN-Form contains the edge information of each CN, which is the same with the information required by the BP algorithm to complete a horizontal process (3). The conversion from the VN-Form to CN-Form adopts a broadcast-based assignment procedure which takes only a little time.

Step 4: Similar to step 2, the horizontal network updates the edge information of each CN. Input the CN-Form into the horizontal network through rows, and the output is the updated edge information with the dimension of $[1, D_c]$. Note that since the updates between multiple VN/CN are independent during the decoding process, all the VN/CN are updated in parallel.

Step 5: Reorganize the updated CN-Form to obtain the VN-Form according to the Tanner graph as show in Fig. 4. Steps 1, 2, 3, 4 and 5 constitute a loop, which corresponds to an iteration in the BP algorithm. In the training phase, we do not perform the parity check to decide whether to stop the iteration. The maximum number of iterations for NNE-BP is fixed as 20. If the maximum number of iterations is not reached, it enters the next loop, otherwise it enters into the output network for judgment. Note that NNE-BP adopts the structure of RNN to imitate the iterative process of the BP algorithm, which means that we adopt the same network weight over multiple iterations.

Step 6: During the training stage, in order to facilitate the calculation of the network loss, we replaced (6) with a classification network to make soft decision on each VN. The network input is consistent with the vertical network, whose output $[p_i^0, p_i^1]$ indicates that the probability of the current VN node is 0/1, respectively. Then the binary cross entropy loss is calculated through (15). In the verification stage, there is no need for Softmax. Instead, directly use $[p_i^0, p_i^1]$ to make a hard decision as follows:

$$\hat{VN}_i = \begin{cases} 1, & p_i^0 < p_i^1 \\ 0, & p_i^1 < p_i^0 \end{cases} \tag{16}$$

C. ANALYSIS OF CHANNEL NOISE

The addition of appropriate noise helps to train a more robust model. In the training process, we add the Additive White Gaussian Noise (AWGN) to channel LLR. For the SNR range of the dataset, we examine by adding Single-SNR (1, 3, 5dB) noise and Mixed-SNR (1–5 dB) noise. The results are presented in Fig. 6. The Single-SNR model ($S-model$) trained on 1dB/3dB reaches the best BER performance with the 1dB/3dB test dataset, while the $S-model$ trained on 5dB performs poorly through all tests. Since the decoding of the LDPC codes is essentially an error correction process, a low SNR noise can provide sufficient error correction experience. However, relying on a single SNR noise may result in the NN falling into a local optimum, which affects its decoding performance in the high SNR range. In contrast, the mixed dataset includes noise blocks among a wide range of SNRs, and also includes an appropriate proportion of error blocks. Consequently, the Mixed-SNR model ($M-model$) demonstrates strong generalization ability across a wide range of SNRs and exhibits the best performance in the high SNR region.

In order to explain the above results more apparently, we separately recorded the vertical network output after

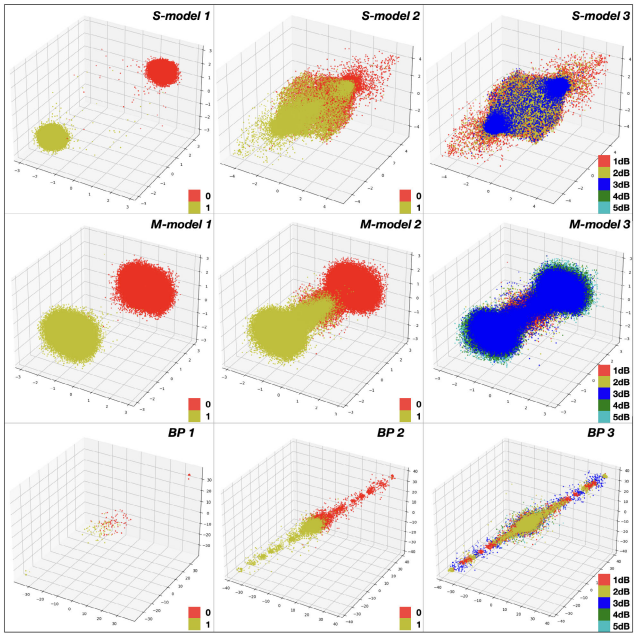


FIGURE 7. Influence of different SNR ranges on intermediate variables of NNE-BP.

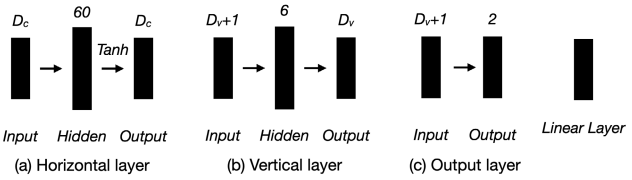


FIGURE 8. Specific structures of sub-networks.

20 iterations of the $S-model$ and $M-model$, i.e., the LLR information carried by the edges connected to each VN before the final decision is made. In the experiment, we select 1000 blocks of samples for each SNR, the degree of VN is 3. Hence, each tensor has three dimensions. We regard the three dimensions as the coordinates of the 3D space, implying that each VN is mapped onto a point in the 3D space as shown in Fig. 7. According to the LLR definition, if the decoding is done correctly, the VN nodes with the values of "0" and "1" should appear in the first and eighth quadrants, respectively. The first and second rows correspond to the $S-model$ and $M-model$, respectively. The third row lists the corresponding data distribution in the BP algorithm for comparison. Each column represents a test environment.

In the first and second columns, color is used to symbolize the label of each VN. In the first column, 5dB data was used for testing. It can be seen that the two models have radically different mappings. $S-model$ maps the data onto two quadrants, but the sample distribution in each quadrant is relatively scattered. The $M-model$ maps the samples of each category onto a spherical space during the classification, which keeps two classes apart and reduce the intra-class gap. This effect plays the same role as the triplet loss mentioned in [6]. A more centralized sample distribution is convenient for the subsequent output layer to make accurate judgement. The performance of the BP algorithm is similar to that of the $M-model$, with majority samples concentrated in a small circle near the edge. In the second column, 1–5 dB mixed data was used for testing. Due to the lack of robustness, the $S-model$ has many misjudgments and it cannot work normally, whereas the $M-model$ can still accurately classify most samples and the samples of each category form two separate circular zones.

In order to further comprehend the phenomenon, we examine the second column from different perspectives. The experimental conditions in the third column are the same with those in column 2, and the five colors correspond to the data with 1–5 dB noise, respectively. It can be seen that in the $M-model$, the data with a high SNR of 4–5 dB is mostly mapped onto the circular region far from the classification surface, which has a large confidence to separate them. However, the low SNR data of 1–2 dB is mostly close to the classification surface and has low confidence, which is in the line with the BP algorithm. The $S-model$ cannot distinguish the data of different SNR values. In the process of decoding, although the data scales between BP and NNE-BP are different, $M-model$ will produce data distribution similar to the traditional BP algorithm, and map the data onto the high and low confidence regions according to SNR. In summary, with the same scale training dataset, the model trained with mixed SNR data will be more generalized, especially in the high SNR range. Therefore, the Mixed-SNR was selected as the noise range of the training data.

D. SELECTION OF NETWORK SCALE

In this section, we discuss the specific parameters and activation function selection of the three sub-networks. Take MacKay (96,48) [21] as an example. The CN and VN degrees of the LDPC codes are 6 and 3, respectively. The horizontal network is used for the update of CN as per (3), which involves a lot of multiplication and logarithm operations. In order to fit this complex nonlinear process, we adopt a hidden layer of a large dimension and put the nonlinear activation function Tanh behind the hidden layer as show in Fig. 8(a). Figure 9 shows the influence of five different dimensions of the hidden layer on the BER performance as the vertical network is fixed.

The vertical network is used to update a single VN through (4). Only a few addition operations are performed in this process. We also maintained the vertical process for replacing the high-complexity horizontal process with neural networks, as the computational cost of simulating simple operations with a neural network might not affected by any loss in efficiency. However, as demonstrated in Fig. 10, the convergence and performance of a local replacement (Horizontal Network+Vertical process) are slightly worse than those of the overall replacement (Horizontal Network+Vertical Network).

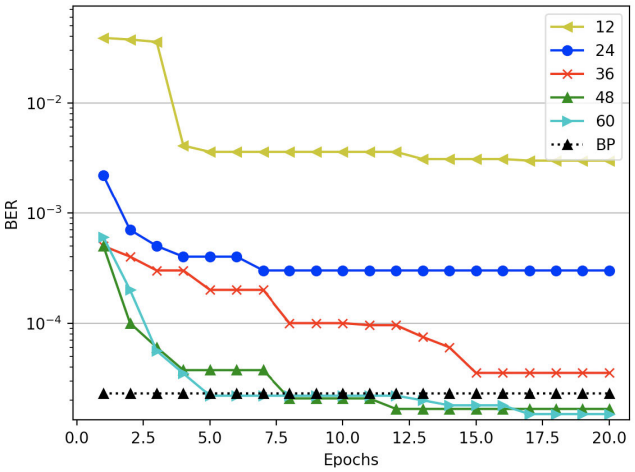


FIGURE 9. Impact of hidden dimension of horizontal layer.

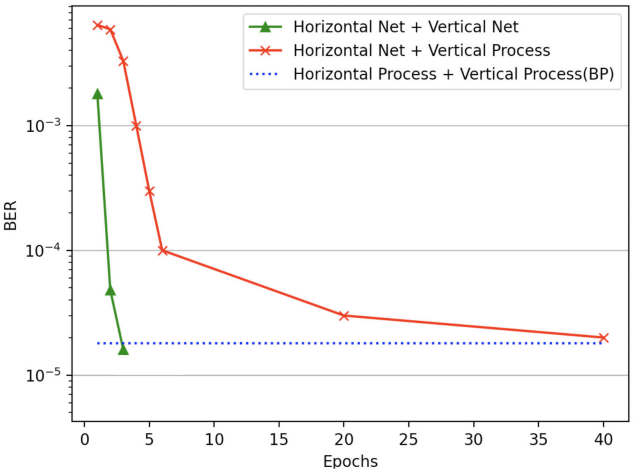


FIGURE 10. Ablation study for vertical network.

TABLE 1. Network scale and SNR range for different LDPC codes

Matrix	N	M	CN	VN	SNR-Range	H-Dim	V-Dim
H1	96	48	6	3	$1-5dB$	60	6
H2	96	32	6	2	$2-6dB$	60	4
H3	112	64	7	4	$0-4dB$	72	8

Finally, a single hidden layer network with a dimension of 6 is used to fit the linear process without any activation function as show in Fig. 8(b).

The output network does not need the hidden layer and activation function, but SoftMax is used to normalize the output when calculating the cross entropy loss in the training phase.

Table 1 shows the minimum network scale modified under other CN&VN degrees obtained from experiments. H-Dim and V-Dim denote the hidden layer dimensions of the horizontal and vertical networks, respectively. An increase in the network scale can speed up the convergence of the training process, but it would not improve the decoding accuracy any further. Furthermore, we opted to utilize separate SNR ranges for each LDPC code, based on its BER performance, in order to ensure that the training dataset contains a reasonable proportion of error blocks. To achieve this, we initially employed the BP algorithm to identify the SNR range corresponding to the BER range of $1*10^{-1}$ to $1*10^{-5}$ for a given LDPC code. Subsequently, we generated the training dataset uniformly within the obtained SNR range.

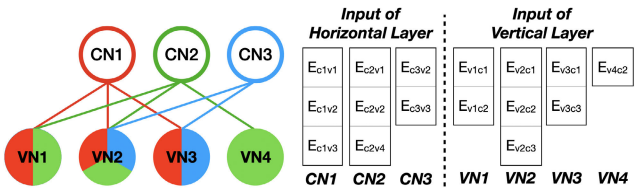


FIGURE 11. An example about irregular LDPC codes.

E. SOLUTION TO IRREGULAR LDPC CODES

In the above discussion, NNE-BP decodes the regular LDPC codes, i.e., All the columns(or rows) in the parity check matrix have the same number of 1's. A regular LDPC code is simple to decode in parallel. But in practical scenarios, an irregular LDPC code with a better BER performance is preferred. For the conventional BP algorithm, where the regularity of an LDPC code will not affect the decoding process. But that of NNE-BP, only the scale of the horizontal/vertical network is proportional to the CN/VN degree. Once the model is fixed, the dimension of the LLR tensor that can be input will also be determined. Only the regular LDPC codes codewords with the required degree of node can be decoded. Take Fig. 11 as an example. A partial Tanner graph of an irregular LDPC code is shown on the left side. The degrees of CN/VN are not fixed. The right side figure lists the edge tensors of each CN and VN, which are the input data of the horizontal and vertical networks, respectively. The dimensions of the edge tensor are not unified, and the neural network cannot receive the input tensors of multiple dimensions at the same time. This factor limits the ability of NNE-BP to decode irregular codewords.

In some deep learning scenarios, we also encounter the problem of mismatching size of the input tensor, such as the size of ROI (region of interest) mismatches in the target detection task [14], [15]. The common method is to resize or pad the image. The resizing method is not suitable for the decoding tasks, which are typically implemented using sampling, but the missing edges cannot be determined according to the surrounding edges like the missing pixels. The application of padding in the decoding of irregular LDPC codes is below covered in detail.

Firstly, consider the feasibility of padding in principle. The horizontal network simulates (3), performs Tanh on all the numbers in the input tensor and then multiplies them. If the padding P_{CN} makes $Tanh(P_{CN}/2) \rightarrow 1$, then the padding will

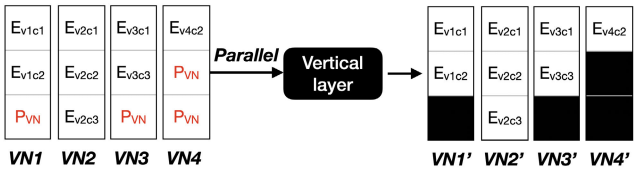


FIGURE 12. Decoding of irregular LDPC codes based on padding scheme.

TABLE 2. Specifications of three irregular check matrices

Matrix	N	M	CN-Distribution	VN-Distribution	P_{CN}	P_{VN}
M1	96	48	5(0.333) 6(0.667)	2(0.167) 3(0.833)	-0.75	-0.02
M2	96	48	6(0.833) 7(0.167)	3(0.917) 4(0.083)	-0.87	-0.06
M3	260	130	5(0.100) 6(0.900)	2(0.050) 3(0.950)	0.77	-0.03

basically not affect the multiplication result. Similarly, the vertical network imulates (4), which adds the input tensor. If the padding P_{VN} is 0, the addition will remain unaffected. However, since NNE-BP is a classification network and the BP algorithm is an accurate mapping process, their data are inconsistent. So, we cannot directly apply the experience gained in the BP algorithm for padding NNE-BP.

We adopt two padding schemes: zero padding and learnable padding. The former fills an irregular CN/VN tensor with zeros as in other vision tasks, while the latter sets the padding-value P_{CN} and P_{VN} as the learnable variables and optimizes them alongside NNE-BP. Taking the vertical network as an example, Fig. 12 shows the VN updating process of the irregular LDPC codes in Fig. 11. In order to begin with, pad the edge vector of each VN to the maximum degree among all the VNs (three in the figure), and then pass it to the corresponding network. After the update, we keep only the real edge information, and the padding edges (black box) are discarded. The padding operation is performed only for data. The network structure and parameter settings in the training process are the same with those used in the regular LDPC codes.

The training process of the irregular LDPC codes takes longer than that of the regular LDPC codes as the optimization of the padding value changes the input data of the network. The decoding network can be updated stably only with a reasonable padding value. Table 2 lists the specifications of three irregular check matrices used in the experiment, as well as the padding-values optimized for them. It can be seen that the padding values of the three matrices are very close to each other, in which all the three values of P_{VN} are close to 0. The absolute values of P_{CN} are relatively large, around 0.8, which are consistent with our previous conjecture based on the BP algorithm.

IV. EXPERIMENTS AND RESULTS

In this section, we will introduce the specific parameter settings used in the training process, compare and analyze the impact of different parameters on the decoding performance,

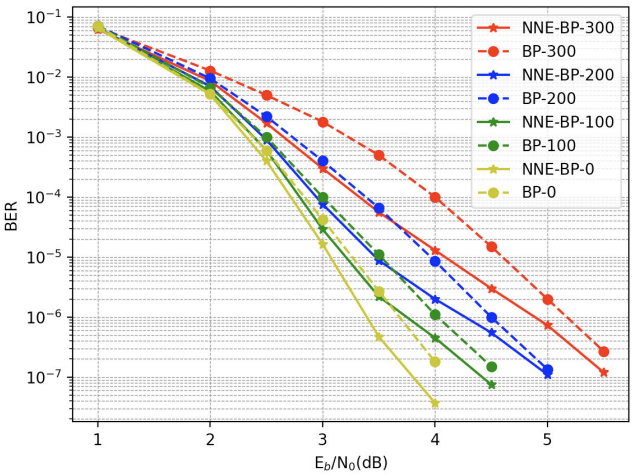


FIGURE 13. Comparison of BER performance with different 4-cycles.

and finally demonstrate the superiority of NNE-BP over the conventional BP algorithm in two aspects: BER performance and decoding efficiency.

A. DATA PREPARATION

We generate the training dataset over the AWGN channel adopted BPSK as the modulation. An equal probability source was used to randomly generate a 0/1 bit stream with a length of K, which was encoded into the (N, K) LDPC codes as the label. After the BPSK modulation, the AWGN noise was added, and it was finally demodulated into LLR, which was the input data of NNE-BP. The training dataset contained a total of $2 * 10^6$ blocks codes, with $4 * 10^5$ blocks codes generated under each SNR among the chosen SNR range. Subsequently, $1 * 10^8$ blocks were generated at each SNR for testing. Without further explanation, all the LDPC codes presented in this paper meet the requirement that the degrees of CN and VN to be 6 and 3, respectively.

B. EXPERIMENTAL CONFIGURATION

We implemented our code in python3.8 on PyTorch1.7, which took about 2 minutes for each epoch training on the entire dataset in a TITAN X GPU with CUDA 10.1 and CuDNN 7.6. Adam [13] was used as the optimizer with a starting learning rate of 0.001 and exponential decay in every five epochs. No warmup was employed, and mini-batches of size 100 blocks were used.

C. ANALYSIS OF BER GAIN

Although the received channel information of BP and NNE-BP is the same, NNE-BP also uses additional source of information during the training process, which is the true label of each VN. Note that the use of more information means the presence of less uncertainty.

On the other hand, the BP algorithm has a fixed decoding process designed under ideal conditions without considering

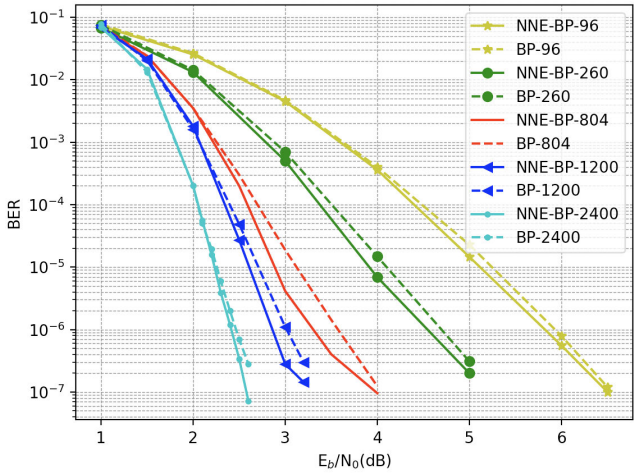


FIGURE 14. Comparison of BER performance under different code lengths.

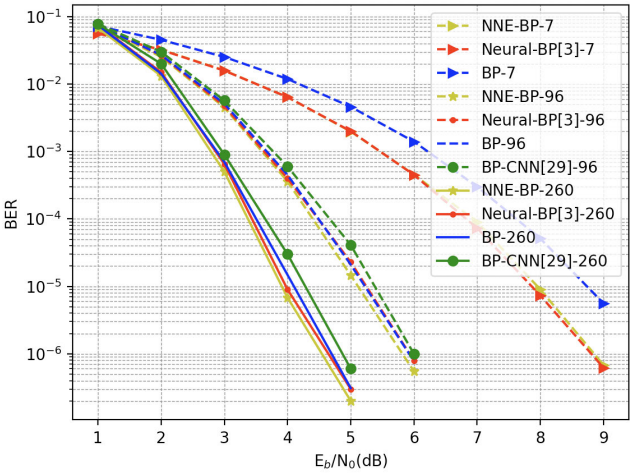


FIGURE 15. Comparison with other DL-based methods.

$\mu's$ impact in (11). So the short-cycles in the tanner graph, especially the 4-cycles, affect it's decoding accuracy. Neural-BP [3] learns different μ for each node. As a result, the network structure is bound to a specific tanner graph and the generalization is lost. We choose a more compromise solution, that is, learning a general μ for all nodes, which not only preserves the generalization, but also can achieve better BER performance than BP algorithm.

To verify this conjecture, we test the gain obtained by NNE-BP on (600, 300) LDPC with different 4-cycles. As shown in the Fig.13, the number after the algorithm indicates the 4-cycles included in LDPC codes. It can be seen that under the same code length, the more 4-cycles LDPC contains, the more obvious gain can NNE-BP obtain, which confirms that NNE-BP plays a role in weakening the influence of short-cycles.

D. GENERALIZATION

Since NNE-BP simulates the most basic node updating process in the BP algorithm, this process is performed for a single node without considering the code length or information transfer. So, the decoding capability of NNE-BP can be extended to all the LDPC codes with the same degree of node via the Tanner graph replacement. We select regular LDPC codes under five code lengths, which are (96, 48), (260, 130), (804, 404), (1200, 602), (2400, 1202). All the check matrices meet the CN/VN degree of 6/3, and the 4-cycles are eliminated. The three LDPC codes which code length is greater than 260 obey the Quasi-Cyclic form and do not contain 6-cycles. Without fine-tuning, the same model trained with MacKay(96, 48) [21] LDPC codes is used to decode the above LDPC codes and its BER performance is comapred with that of the BP algorithm by testing on $1 * 10^8$ blocks. Figure 14 shows the comparison result. Under the listed five code lengths, the BER performance of NNE-BP is slightly better than that of the BP algorithm.

E. COMPARISON WITH OTHER DL-BASED METHODS

We compare NNE-BP with neural-BP [3] and BP-CNN [28]. The network structure of neural-BP is set to 20 *odd layers* + 20 *even layers*, which corresponds to 20 full BP iterations; The structure of BP-CNN is set to $BP(10) + CNN + BP(10)$, and the correlation coefficient η is set to 0. Because Neural-BP is difficult to train on long block-length codes, we choose BCH (7,4), LDPC (96, 48) and LDPC (260, 130) three short codewords to test, and the results are shown in Fig. 15. On the BCH code with more 4-cycles, NNE-BP and neural-BP achieve significant gains over BP algorithm. On the LDPC code without 4-cycles, the performance gains of NNE-BP and neural-BP are reduced. While BP-CNN is designed for correlated noise channel, so its performance is worse than BP under AWGN channel.

F. DECODING OF IRREGULAR LDPC CODES

Figure 16 shows the BER performance of NNE-BP on the irregular LDPC codes listed in Table 2. Since the BER curves of M1 and M2 overlap heavily, only M1 is presented here. In comparison to the BP algorithm, NNE-BP still has a BER gain. However, the irregularities of the above three matrices are small, and only one or two position is needed for padding each CN/VN tensor. In further experiments, we found that for some LDPC codes with large irregularities, NNE-BP no longer get BER gain or even BER performance lags behind that of the BP algorithm. This might due to the fact that the padding of the LLR tensor inevitably introduces some incorrect information, even if the padding value is the optimal one. The more the irregularity of the check matrix, the more padding is required to follow more errors, which is unacceptable for high-precision classification problems like decoding.

G. DECODING EFFICIENCY

1) DECODING UNDER FIXED NUMBER OF ITERATIONS

In this section, we compare the time consumption of the BP, MS, and NNE-BP algorithms under the same BER perfor-

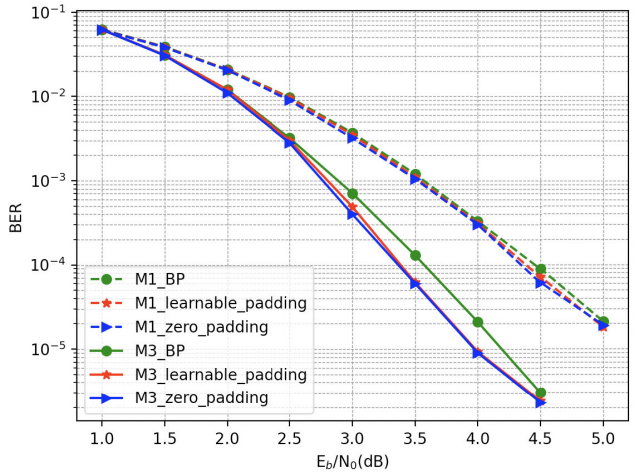


FIGURE 16. Comparison of BER performance under irregular LDPC codes.

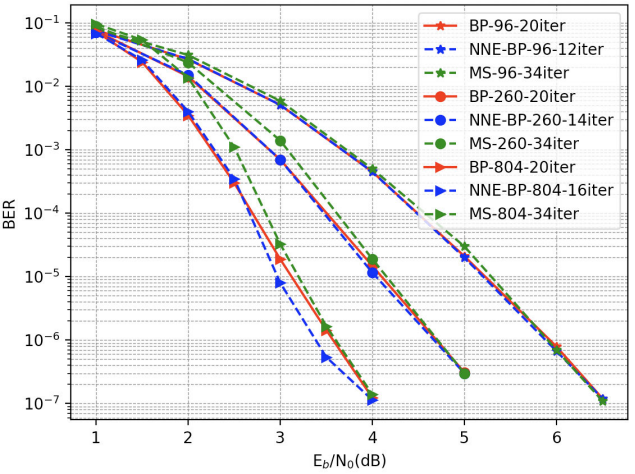


FIGURE 17. Comparison of BER performance under different iterations.

mance. Both algorithms are implemented in Python, and all the VN/CN calculations are performed in parallel. As shown in Fig. 17, we appropriately reduced the number of iterations of NNE-BP to achieve the BER performance similar to that of the BP algorithm. When the code length is 96, eight iterations could be reduced; and when the code length is up to 804, only four iterations could be reduced. It can be seen that with the increase in the code length, the number of iterations that can be reduced by NNE-BP gradually decreases. We find it difficult to fully align the BER curves of BP algorithm and MS algorithm, so we only align their BER at $1 * 10^{-7}$. Finally iterations of the MS algorithm is set to 34. We select the intermediate length of 260 for comparing the subsequent decoding efficiencies.

The amount of required calculations is significantly higher in NNE-BP than in the BP algorithm. However, in terms of the parallelism, the majority computation in the NNE-BP are matrix multiplication with high parallelism. However, the BP algorithm contains more serial operations and logarithmic operations, resulting in less parallelism. We conduct tests with (260, 130) LDPC codes on four different devices by taking test samples of $1 * 10^5$ blocks. Table 3 shows the average results of 10 repeated experiments.

TABLE 3. Comparison of decoding efficiencies in different devices

Devices	Core	Clock speed	Batch	Net/s	BP/s	MS/s	Gains
Core i5	2	2.90GHz	1	639.5	564.2	512.1	-13.3%
Ryzen 5800X	8	2.90GHz	1	341.1	422.1	387.1	19.2%
Xeon Gold 6326	16	2.90GHz	1	163.6	242.2	218.2	32.5%
TiTan X	3584	2GHz	20	54.8	92.78	82.1	40%
			30	35.7	59.4	53.4	39.9%
			40	28.2	46.6	42.7	39.7%
A40	10752	1740MHz	20	15.8	26.4	25.0	40.4%
			30	10.9	18.4	17.3	40.7%
			40	8.2	13.7	12.7	40.1%

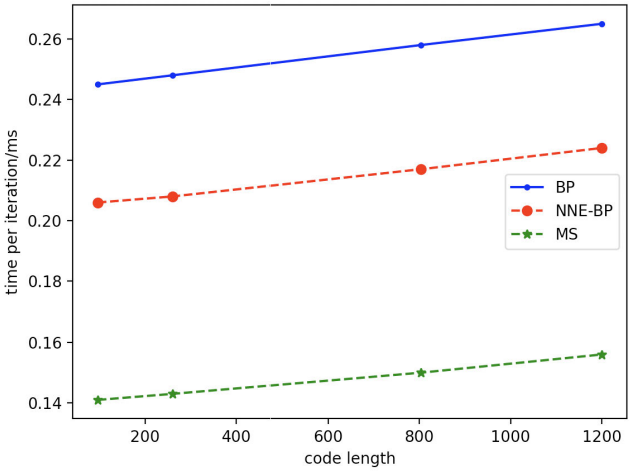


FIGURE 18. Average time consumed per iteration of the three algorithms under different code lengths.

The CPU is used to simulate the devices with limited parallelism. We constrain the clock speed of all CPUs to 2.9GHZ, and due to the weak parallel ability of the CPU, we only decode one code block at a time. It can be seen that on the dual-core CPU, the time consumption of NNE-BP is much higher than that of the BP algorithm as the bottleneck of the decoding efficiency at this time lies with the computation power. The algorithm with less computation obtains higher throughput. NNE-BP gradually gains efficiency as the number of CPU cores increases. The time gain of NNE-BP reached 32.5% on a 16-core CPU. What GPU simulates is a device with adequate computing power. It can be seen that the decoding time and parallel block batches basically present a linear inverse relationship, indicating that the device has sufficient computing resources. At this point, the parallelism

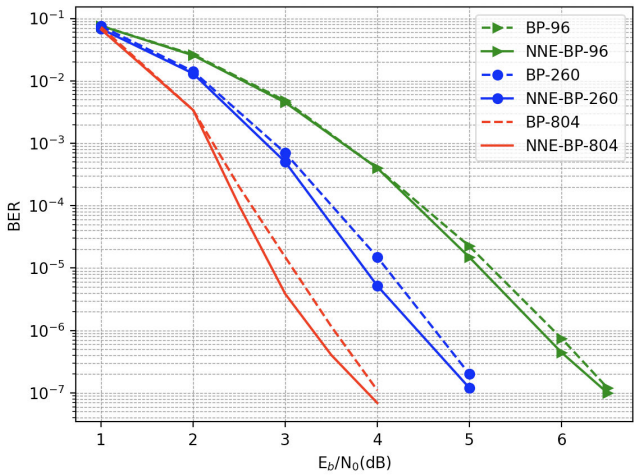


FIGURE 19. **Comparison of BER performance under parity check.**

TABLE 4. **Comparison of average iterations under parity check**

Length	Method	1dB	2dB	3dB	4dB	5dB
96	NNE-BP	**15.54**	**9.74**	**4.88**	**2.66**	**1.74**
	BP	15.55	9.8	5.2	2.85	1.83
260	NNE-BP	17.8	10.7	**5.26**	**3.13**	**2.18**
	BP	**17.7**	**10.6**	5.4	3.33	2.31
804	NNE-BP	19.44	10.71	5.91	**3.75**	**2.64**
	BP	**19.37**	**10.67**	**5.83**	3.91	2.83

of the algorithm becomes the only factor to determine the decoding efficiency, and the time gain obtained by NNE-BP is basically maintained at around 40%.

In order to prove that our model does not become complicated with the increase in the code length, we test the average time required for each iteration under four code lengths and compare it with that of the BP algorithm and MS algorithm on GPU device. As shown in Fig. 18, under each code length, NNE-BP reduces the time consumed in each iteration by 20% compared with that of the BP algorithm. Moreover, when the code length increases from 96 to 1200, the average iteration time of NNE-BP increases linearly by 8.1%, which is lower than 8.8% of the BP algorithm.

2) DECODING WITH PARITY CHECK

In this section, we no longer fix the number of iterations of decoding, but conduct parity check after each iteration to check whether $H \cdot \hat{x}^T = 0$ is satisfied. The maximum number of iterations is set to 20. If the check is successful, the decoding is terminated in advance. Table 4 compares the average iterations of the two algorithms under various channel conditions, where NNE-BP can slightly reduce the number of iterations in the high SNR region. Figure 19 presents the BER comparison of the two algorithms in the case of parity check, where NNE-BP still maintains its advantage.

V. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In this paper, we propose a novel neural network decoder, named as NNE-BP. The proposed framework uses a small-scale neural network to replace the most basic node updating operation of the BP algorithm. The network architecture is not restricted to any particular Tanner graph or code length. It significantly reduces the number of parameters when improving the generalization in comparison with the existing DL decoders. Compared with the traditional BP algorithm, the parallelism of the decoding process is improved and the efficiency gain is obtained in a high performance computing equipment. Additionally, the BER performance is improved due to the weakening of the short-cycle effects.

There are still many aspects worth investigating in NNE-BP. For example, the current padding scheme for irregular codes does not perform well when dealing with LDPC codes with high irregularity like 5G-LDPC codes. Besides, the simple network structure we adopted can not handle some nodes in tanner graph with high degree (such as 20) well. We will seek for a better solution in our future research. Furthermore, as a neural network alternative to the BP decoding algorithm, the application scope of NNE-BP should not be limited to LDPC codes only. We will seek its application to other linear codes.

ACKNOWLEDGMENT

(Die Hu and Jun Wu contributed equally to this work.)

REFERENCES

[1] Q. Wang, S. Wang, H. Fang, L. Chen, L. Chen, and Y. Guo, "A model-driven deep learning method for normalized min-sum LDPC decoding," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[2] T. Gruber, S. Cammerer, J. Hoydis, and S. T. Brink, "On deep learning-based channel decoding," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2017, pp. 1–6.

[3] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[4] X. Ding, J. An, Z. Zhao, X. Bu, and K. Yang, "Low-density parity-check coded direct sequence spread spectrum receiver based on analog probabilistic processing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 7, pp. 6355–6370, Jul. 2021.

[5] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2016, pp. 341–346.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[7] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[8] X. Pang, C. Yang, Z. Zhang, X. You, and C. Zhang, "A channel-blind decoding for LDPC based on deep learning and dictionary learning," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2019, pp. 284–289.

[9] Q. Wu, S. Tang, Y. Liang, C. T. Lam, and Y. Ma, "A low complexity model-driven deep learning LDPC decoding algorithm," in *Proc. IEEE 6th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Apr. 2021, pp. 558–563.

[10] N. Shah and Y. Vasavada, "Neural layered decoding of 5G LDPC codes," *IEEE Commun. Lett.*, vol. 25, no. 11, pp. 3590–3593, Nov. 2021.

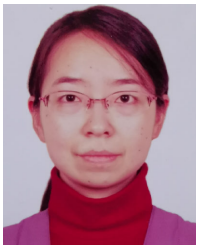
[11] T. Watanabe, T. Ohseki, and K. Yamazaki, "Deep learning-based bit reliability based decoding for non-binary LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1451–1456.
[12] J. Dai et al., "Learning to decode protograph LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1983–1999, Jul. 2021.
[13] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–16.
[14] V. Papyan and M. Elad, "Multi-scale patch-based image restoration," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 249–261, Jan. 2016.
[15] F. Hu and X. Zhu, "Auroral image super-resolution via dynamic nonlocal intensity fusion and low-rank regularization," in *Proc. 3rd Int. Conf. Natural Lang. Process. (ICNLP)*, Mar. 2021, pp. 232–236.
[16] X. Jiao et al., "l_2-Box ADMM decoding for LDPC codes over ISI channels," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3966–3971, Mar. 2021.
[17] H. Lam, S. Lu, H. Qiu, M. Zhang, H. Jiao, and S. Zhang, "A high-efficiency segmented reconfigurable cyclic shifter for 5G QC-LDPC decoder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 1, pp. 401–414, Jan. 2022.
[18] X. Zhang and S. Chen, "A two-stage decoding algorithm to lower the error-floors for LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 4, pp. 517–520, Apr. 2015.
[19] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[20] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406–414, Mar. 2002.
[21] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[22] T. M. N. Ngatched and F. Takawira, "Improved generalized low-density parity-check codes using irregular graphs," *Trans. South Afr. Inst. Electr. Eng.*, vol. 94, no. 4, pp. 43–49, Dec. 2003.
[23] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of LDPC codes and extension to turbo decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, Mar. 2001, p. 189.
[24] T. Liu and X. Chen, "Deep learning-based belief propagation algorithm over non-binary finite fields," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2020, pp. 164–169.
[25] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," in *Proc. Inf. Theory Workshop*, 1998, pp. 70–71.
[26] D. Oh and K. K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 105–115, Jan. 2010.
[27] H. Zhu, Z. Cao, Y. Zhao, and D. Li, "Learning to denoise and decode: A novel residual neural network decoder for polar codes," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8725–8738, Aug. 2020.
[28] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 144–159, Feb. 2018.
[29] J. L. Kim, U. N. Peled, I. Perepelitsa, V. Pless, and S. Friedland, "Explicit construction of families of LDPC codes with no 4-cycles," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2378–2388, Sep. 2004.
[30] X. Tao, Y. Li, Y. Liu, and Z. Hu, "On the construction of LDPC codes free of small trapping sets by controlling cycles," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 9–12, Jan. 2018.

HAO XU received the B.S. degree in information security from Xidian University, Xi'an, China, in 2018, and the M.S. degree in computer science and technology from Ningxia University, Yinchuan, China, in 2020. He is currently pursuing the Ph.D. degree in computer science and technology with Tongji University, Shanghai, China. His research interests include multimedia communication and machine learning.

YING LI received the B.S. degree in computer science and technology from Shanghai Normal University, Shanghai, China, in 2016. She is currently pursuing the Ph.D. degree in computer science and technology with Tongji University, Shanghai. Her research interests include multimedia communication and wireless networking.

BIN TAN received the Ph.D. degree in computer science and technology from Tongji University, Shanghai, China, in 2017. She is currently a Professor with the College of Electronics and Information Engineering, Jinggangshan University. Her current research interests include multimedia communication, source and channel coding, and wireless networking.

JUN WU (Senior Member, IEEE) received the B.S. degree in information engineering and the M.S. degree in communication and electronic systems from Xidian University, Xi'an, China, in 1993 and 1996, respectively, and the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications, Beijing, China, in 1999. He was a Professor with the Department of Computer Science and Technology, Tongji University. He was also a Principal Scientist with Broadcom Inc., before he joined Tongji University. He is currently a Full Professor with the School of Computer Science, Fudan University. His research interests include wireless networks, machine learning, and signal processing.



DIE HU received the B.S. and Ph.D. degrees in electronic engineering from Southeast University, Nanjing, China, in 2001 and 2006, respectively. She is currently an Associate Professor with the Department of Communication Science and Engineering, Fudan University, Shanghai, China. Her research interests include wireless communications and signal processing.