Physical Communication 61 (2023) 102194

Contents lists available at ScienceDirect

Physical Communication

journal homepage: www.elsevier.com/locate/phycom

Full length article

Neural layered min-sum decoders for cyclic codes

Ming Wang [a], Yong Li [a], Jianqing Liu [b], Taolin Guo [a,*], Huihui Wu [c], Francis C.M. Lau [d]

[a] College of Computer Science, Chongqing University, Chongqing, 400044, China
[b] Department of Computer Science, North Carolina State University, Raleigh, 27606, NC, USA
[c] Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, Canada
[d] Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong, China

ARTICLE INFO

Article history:
Received 24 May 2023
Received in revised form 30 July 2023
Accepted 11 September 2023
Available online 17 September 2023

Dataset link: https://github.com/Dies-Irae/Neural-Layered-Min-sum-Decoders-for-Cyclic-Codes

Keywords:
Channel codes
Neural network decoders
Layered min-sum algorithm
Modified random redundant decoding

ABSTRACT

This paper proposes a low-complexity neural network decoder based on the layered min-sum algorithm to decode cyclic codes. By generalizing the layered min-sum algorithm to its neural network counterpart, the number of network weights decreases while retaining a good error correction performance. The Bose–Chaudhuri–Hocquenghem (BCH) codes, quadratic residue (QR) codes, and punctured Reed–Muller (RM) codes are selected as three exemplary binary cyclic codes. Simulation results show that the proposed neural decoder achieves superior performance with less computational complexity compared with the state-of-the-art neural network decoder. Further, a neural decoder incorporating the modified random redundant decoding (mRRD) algorithm is investigated to approach the performance of maximum-likelihood decoding for some short codes.

1. Introduction

Cyclic codes such as Bose–Chaudhuri–Hocquenghem (BCH) codes and quadratic residue (QR) codes are important channel codes used in physical-layer digital communications. For example, BCH codes have been applied in satellite communications [1] and Compact Disc players. The (24,12) extended Golay code has been used in the imaging systems for space exploration [2] and high-frequency radio systems [3]. Recently, short cyclic codes have returned to the spotlight for their good decoding performance under maximum-likelihood (ML) approaching decoding algorithms [4]. However, extensive computations are required to attain such a good decoding performance. Taking the BCH(63,45,7) code as an example, it needs up to $50 \times 50 \times 5$ iterations to achieve the near ML performance when the modified random redundant decoding (mRRD) [5] algorithm with neural BP decoding is employed [6]. Accordingly, efficient decoders for these codes have been in high demand, especially for achieving near ML performance.

Due to the great success of AI in various fields, the idea of using machine learning to design or decode channel codes has been extensively investigated in recent years. In [7], the authors showed that recurrent neural networks (RNN) can be trained to decode convolutional codes and turbo codes, and in [8–10], data-driven end-to-end autoencoders have been utilized to learn to encode and decode simultaneously. Huang et al. [11] used reinforcement learning and genetic algorithm to design some binary codes, which achieve equivalent performance to some existing codes and even outperform certain codes constructed by traditional methods. An important branch of these studies is the neural belief propagation (BP) based decoders. It is known that BP-based decoders can achieve near-capacity performance for decoding low-density parity-check (LDPC) codes and it also has been used in the optimization of Gaussian multiple access channel (GMAC) [12]. However, the performance of BP-based decoders deteriorates significantly for decoding some cyclic codes such as BCH and QR codes since the parity-check matrices of these codes are much denser. As an iterative message passing algorithm, the BP-based algorithms can be naturally generalized to neural networks. In [13], a neural BP decoder was introduced. It assigns different weights to each edge of the Tanner graph and trains these weights, and greatly outperforms the conventional BP algorithm for decoding BCH codes. Further, the complexity of neural BP decoder is reduced in [6,14] by replacing the sum–product operations with min-sum functions. The authors also proposed using additive weights instead of multiplicative weights to further lower the computational complexity. In [15], the authors proposed a neural BP-based scalable decoder of which parameters are determined by full-connected neural networks.

* Corresponding author.
E-mail addresses: mwang42@ncsu.edu (M. Wang), yongli@cqu.edu.cn (Y. Li), jliu96@ncsu.edu (J. Liu), tguo@cqu.edu.cn (T. Guo), huihui.wu@mail.mcgill.ca (H. Wu), francis-cm.lau@polyu.edu.hk (F.C.M. Lau).

https://doi.org/10.1016/j.phycom.2023.102194
1874-4907/

The benefit is that the number of weights is independent of code length and the network trained on short codes can be easily scaled to long codes. In [16], the authors modified the activation function of nodes to maintain the sparse property. The authors also proposed a method to learn from the teacher decoder, which is a decoder with better performance. The proposed algorithm improves the error performance while the complexity is the same as the neural BP decoder. Recently, the authors of [17] proposed a cyclically equivariant neural decoder to decode cyclic codes. They exploited the cyclic property of the codes to reduce the number of trainable weights, and the decoding procedure is performed on cyclic redundant parity-check matrices. The cyclically equivariant neural decoder achieves better error correction performance than the decoders proposed in [6,18]. Additionally, the authors in [17] proposed a list decoding method to improve the performance of cyclically equivariant decoder. Particularly, it requires up to $64 \times 20\times 5$ iterations to achieve the near ML performance of the BCH(63,45,7) code. This is much less than that of the mRRD decoder in [6] which otherwise requires up to $50 \times 50 \times 5$ iterations. Buchberger et al. [19] proposed to perform decoding on large redundant matrices to obtain better performance. In order to construct these huge parity-check matrices, the authors used all the codewords with the minimum weight from the dual codes of original codes as rows. Next, they trained neural networks representing these matrices and pruned the rows and the corresponding neural network at each iteration. The proposed decoder achieves near ML performance on certain Reed–Muller codes without using the list decoder or mRRD algorithm. However, the number of codewords with the minimum weight may be numerous in the dual code, and it is even difficult to calculate the minimum distances of some codes. We also point out that the computational complexity at each iteration increases with the growing dimension of parity-check matrices for the BP-based algorithms. In contrast, [20] proposed a bottom-up neural min-sum method which introduces trainable parameters gradually to decode LDPC codes. This method allows better tradeoff between complexity and performance. Neural BP-based algorithms have also been utilized to decode quasi-cyclic (QC) LDPC codes [21].[1]

It is noted that the layered min-sum algorithm [22] enjoys much faster convergence compared with the flooding scheduled version. This motivates us to construct neural networks based on the layered min-sum algorithm, so as to further reduce the computational complexity and enhance the decoding performance. To summarize, the main contributions of this paper include the following aspects.

- We propose a low complexity neural decoder based on the layered min-sum algorithm. The proposed decoder achieves better performance using the same number of iterations while fewer trainable weights compared to existing neural decoders. It also supports that neural BP-based decoders achieve better performance by attenuating the magnitude of messages rather than compensating for short cycles.
- We apply the mRRD algorithm to the proposed decoder to obtain near ML performance for some codes, and we introduce an early stopping criterion to accelerate the decoding process. The average number of iterations needed for different signal-to-noise ratios (SNR) are significantly lower than that of the list decoder in [17] and the maximum number of iterations is merely 1/5 of the latter.

[1] The algorithm assigns weights by exploiting the quasi-cyclic structure of QC-LDPC codes, which makes it inapplicable for common cyclic codes.

2. Preliminaries

An (n, k, d) linear block code is a code of length n, dimension k and minimum Hamming distance d. Assume the communication channel is a binary input additive white Gaussian noise (BI-AWGN) channel. A codeword $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in \{0, 1\}^n$ is first modulated as $\mathbf{y} = ((-1)^{u_1}, (-1)^{u_2}, ..(-1)^{u_n})$ and then $\mathbf{y}$ is sent over the channel. Let the noise vector be $\mathbf{n} \in \mathbb{R}^n$, where $n_i \sim \mathcal{N}(0, \sigma^2)$. Then the received vector is $\mathbf{r} = \mathbf{y} + \mathbf{n}$. In the flooding scheduled min-sum algorithm, the log-likelihood ratios (LLRs) are passed simultaneously and repeatedly between the check nodes and variable nodes, and the update rules at the kth iteration can be described as follows.

1. Variable nodes v_i to check nodes c_j update:
$$l^k_{v_i\to c_j} = l^{ch}_{v_i} + \sum_{c_{j'}\in N(v_i)\setminus\{c_j\}} l^k_{c_{j'}\to v_i}, \tag{1}$$
where $l^{ch}_{v_i}$ is the channel LLR of v_i and $N(v_i)$ refers to the check nodes that are adjacent to node v_j and "\" is a subtraction between two sets.
2. Check nodes c_j to variable nodes v_i update:
$$l^{k+1}_{c_j\to v_i} = \min_{v_{i'}\in N(c_j)\setminus\{v_i\}} |l^k_{v_{i'}\to c_j}| \cdot \prod_{v_{i'}\in N(c_j)\setminus\{v_i\}} \operatorname{sign}(l^k_{v_{i'}\to c_j}). \tag{2}$$

To perform a min-sum algorithm with i_{max} iterations, the above updates are repeated for i_{max} times. In the neural normalized min-sum (NNMS) algorithm [6], Eq. (2) is modified as
$$l^{k+1}_{c_j\to v_i} = w^k_{v_{i'}\to c_j} \min_{v_{i'}\in N(c_j)\setminus\{v_i\}} |l^k_{v_{i'}\to c_j}| \cdot \prod_{v_{i'}\in N(c_j)\setminus\{v_i\}} \operatorname{sign}(l^k_{v_{i'}\to c_j}), \tag{3}$$
where $w_{v_i\to c_j} \in (0, 1)$ is the trainable weight of the edge (v_i, c_j).

In contrast to the flooding min-sum algorithm, the layered (or asynchronous) min-sum algorithm converges much faster while retaining the same or better error correction performance. The reason is that the check nodes are not updated simultaneously but sequentially in the layered min-sum algorithm. Therefore, the check nodes updated later could utilize the latest LLR information from the previously updated nodes in the same iteration. This motivates us to generalize the layered min-sum algorithm to its neural network counterpart as described next.

3. The proposed neural decoder

3.1. Neural layered min-sum

The layered min-sum algorithm divides the Tanner graph into several clusters and serially performs decoding iterations within each cluster. Thus, nodes can acquire newer information from priorly updated nodes and accelerate the convergence. In addition to generalizing neural min-sum algorithm to its layered counterpart, we will explore a different policy of weight assignment. BP-based algorithms usually have unsatisfactory performance on codes with dense parity-check matrices, and short cycles are commonly considered as the main reason for this phenomenon [6,13,15,19]. As a result, edge-wise weight assignment are proposed to compensate the effect of short cycles by making messages in short cycles less reliable. However, according to [23], the edge-wise weighted neural BP decoder do not significantly change the distribution of errors, and in some circumstances, small cycles are not the main factor of performance degradation. Which means the performance gain of neural BP-based decoders may not be achieved by compensating short cycles or trapping sets. Therefore, we set iteration-wise weights for the neural layered min-sum decoder to reduce the number of weights, while

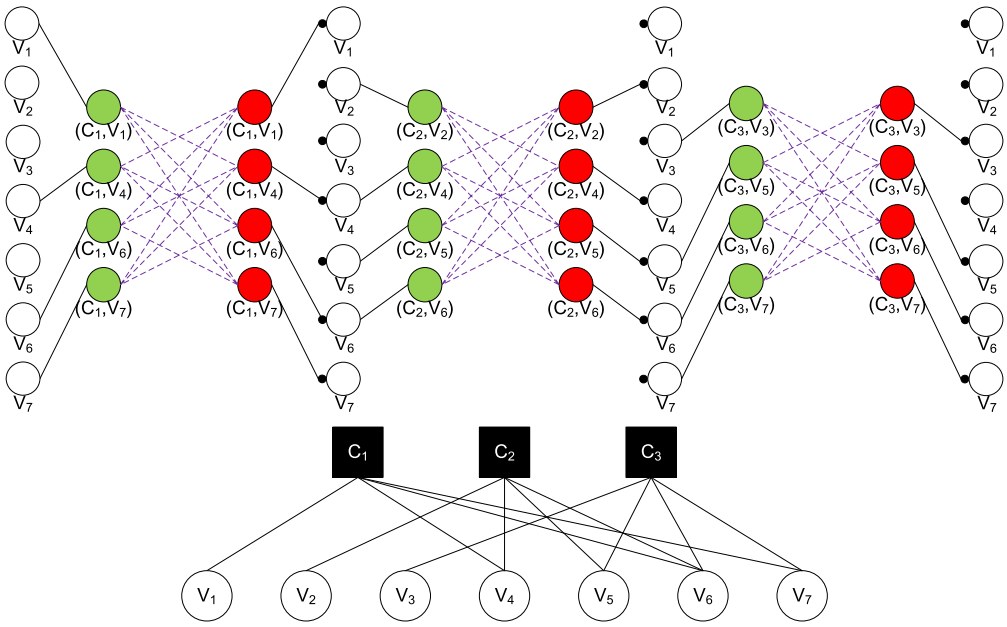


Fig. 1. The exemplary structure of the proposed neural layered min-sum decoder in accordance to the Tanner graph at the bottom. The purple dashed lines correspond to trainable weights, and it is noted that they share the same values at each iteration. The black circles represent the additions of LLRs from the previous update.

still retaining the error correction performance (see Fig. 2). The reduction on trainable weights is significant. For example, an $(n-k)\times n$ parity-check matrix of the (63,45,7) BCH code has $n_e = 432$ edges and row weight $n_r = 24$. If we use edge-wise weights as in the NNMS algorithm with i_{max} iterations, the total number of weights would be $n_e \times i_{max} = 432 \times i_{max}$. The cyclically equivariant decoder (CEBP) [17] requires an $n \times n$ cyclic redundant matrix and needs $n_r \times n_r \times i_{max} = 576 \times i_{max}$ weights. The edge-wise weighted NNMS would need $n \times n_r \times i_{max} = 1512 \times i_{max}$ weights using cyclic redundant matrix. The original matrix $\mathbf{H}$ and cyclic redundant matrix $\mathbf{H}_{cyc}$ have the form of Eq. (4), where H_1 is the first row of $\mathbf{H}$ and "$\gg$" is cyclically right shift.

$$\mathbf{H} = \begin{bmatrix} H_1 \\ H_1 \gg 1 \\ \cdots \\ H_1 \gg (n-k-1) \end{bmatrix}, \mathbf{H}_{cyc} = \begin{bmatrix} \mathbf{H} \\ H_1 \gg (n-k) \\ \cdots \\ H_1 \gg (n-1) \end{bmatrix} \tag{4}$$

On the other hand, if we employ the iteration-wise weights, the total number of weights would be only i_{max} regardless of the utilization of decoding matrices. The update rule for each iteration in the neural layered min-sum algorithm is described as follows.

For each check node c_j, variable nodes $v_i \in N(c_j)$, and $1 \le k \le i_{max}$:

1. Variable nodes to check nodes update, where $l^1_{v_i} = l^{ch}_{v_i}$ and $l^0_{c_j \to v_i} = 0$:
$$l^k_{v_i} = l^k_{v_i} - l^{k-1}_{c_j \to v_i}. \tag{5}$$
2. Check nodes to variable nodes update:
$$l^k_{c_j \to v_i} = w^k \min_{v_{i'} \in N(c_j)\setminus\{v_i\}} |l^k_{v_{i'}}| \prod_{v_{i'} \in N(c_j)\setminus\{v_i\}} \text{sign}(l^k_{v_{i'}}). \tag{6}$$
3. Variable nodes LLR update:
$$l^{k+1}_{v_i} = l^k_{v_i} + l^k_{c_j \to v_i} \tag{7}$$

By replacing $w^k_{v_{i'} \to c_j}$ in Eq. (3) by w^k in Eq. (6), we greatly reduce the number of weights.

The structure of the neural layered min-sum algorithm is demonstrated in Fig. 1. The output vector $\mathbf{o}$ can be interpreted as an estimation of the LLR vector, i.e. $o_i \approx \log \frac{P(u_i=0|\mathbf{r})}{P(u_i=1|\mathbf{r})}$. Therefore, the output can be converted to the probability estimation from the sigmoid function ($\text{sig}(x) = 1/(1+e^{-x})$). In other words, $\text{sig}(o_i) \approx P(u_i = 0|\mathbf{r})$ and $\text{sig}(-o_i) \approx P(u_i = 1|\mathbf{r})$. In order to measure the difference between the output vector $\mathbf{o}$ and the labels (i.e., the true codeword $\mathbf{u}$), a binary cross entropy loss function is used, i.e.,

$$L(\mathbf{o}, \mathbf{u}) = -\frac{1}{n}\sum_{i=1}^{n} u_i \log(\text{sig}(-o_i)) + (1-u_i)\log(\text{sig}(o_i)). \tag{8}$$

3.2. Accelerated neural mRRD

To further enhance the decoding performance of the proposed neural layered min-sum algorithm, the mRRD algorithm is incorporated. It permutes the received vector multiple times and then decodes these permuted vectors. A simple example of permutation is described as follows. Given a permutation $\pi = \left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{smallmatrix}\right)$, it permutes any codeword $\mathbf{u} = (u_1, u_2, u_3)$ to $\pi(\mathbf{u}) = (u_3, u_2, u_1)$. The permutations used in the decoding process are chosen from the automorphism group $\text{Per}(\mathcal{C})$, which consists of all permutations that preserve the codewords from the code ensemble $\mathcal{C}$. The automorphism group is defined by

$$\text{Aut}(\mathcal{C}) = \{\pi \,|\, \pi(\mathbf{u}) \in \mathcal{C}, \forall \mathbf{u} \in \mathcal{C}\}. \tag{9}$$

We also apply an early stopping criterion [24,25] to the mRRD algorithm, and it enables the decoding procedure to stop immediately once the ML codeword is found. The mRRD algorithm of size $(\mathcal{W}, \mathcal{L}, i_{max})$ is summarized in **Algorithm** 1. The size of a decoder is denoted by $(\mathcal{W}, \mathcal{L}, i_{max})$, implying the required number of iterations in the worst case is $\mathcal{W} \times \mathcal{L} \times i_{max}$. The core idea of the mRRD algorithm is to permute the input vector, and thus the input diversity grows greatly. As a consequence, the application of permutation may yield different outputs and increases the probability of producing a correct result. Moreover, permuting the codewords during the decoding procedure can effectively prevent the propagation of false information caused by trapping sets or short cycles [23]. Finally, we can choose the best estimation from

all the successfully decoded results in terms of their correlation discrepancies $\lambda(\mathbf{r}, \hat{\boldsymbol{u}})$. Herein, a threshold $G_T(\hat{\boldsymbol{u}}, d)$ [25] defined to be the sum of magnitude of some components from $\mathbf{r}$ is used to check whether a decoded vector is an ML codeword, where $\hat{\boldsymbol{u}}$ is a decoding result. The details of this early stop criterion is described as follows. Firstly we define

$$z_i = \begin{cases} 0 \text{ if } r_i \geq 0 \\ 1 \text{ if } r_i < 0 \end{cases}, \tag{10}$$

$$D_0(\hat{\boldsymbol{u}}) = \{i \,|v_i = z_i\}, \quad D_1(\hat{\boldsymbol{u}}) = \{i \,|v_i \neq z_i\}, \tag{11}$$

$$\lambda(\mathbf{r}, \hat{\boldsymbol{u}}) = \sum_{i \in D_1(\hat{\boldsymbol{u}})} |r_i|. \tag{12}$$

Assume there are $n_{\hat{\boldsymbol{u}}}$ elements in $D_1(\hat{\boldsymbol{u}})$, and the index set $D_0(\hat{\boldsymbol{u}})$ has $n - n_{\hat{\boldsymbol{u}}}$ elements, which can be arranged in ascending order of reliability. That is, $D_0(\hat{\boldsymbol{u}}) = \{l_1, l_2, \ldots, l_{n-n_{\hat{\boldsymbol{u}}}}\}$ and $|r_{l_i}| \leq |r_{l_j}|$ for $i < j$. The first j elements in $D_0(\hat{\boldsymbol{u}})$ is denoted as:

$$D_0^{(j)}(\hat{\boldsymbol{u}}) = \{l_1, l_2, \ldots, l_j\}. \tag{13}$$

Define $\delta = d - n_{\hat{\boldsymbol{u}}}$ and $G_T(\hat{\boldsymbol{u}}, d) = \sum_{i \in D_0^{(\delta)}(\hat{\boldsymbol{u}})} |r_i|$. If $\lambda(\mathbf{r}, \hat{\boldsymbol{u}}) \leq G_T(\hat{\boldsymbol{u}}, d)$, then $\hat{\boldsymbol{u}}$ is the ML codeword of $\mathbf{r}$.

Algorithm 1 The proposed mRRD($\mathcal{W}$, $\mathcal{L}$, i_{max})

Input: Received vector $\mathbf{r}$
Output: Decoded vector $\hat{\boldsymbol{u}}$
Initialize: $\mathbf{o} = \mathbf{r}$, $\mathbb{S} = \emptyset$
for $i \in \{1, \ldots, \mathcal{W}\}$ **do**
  for $j \in \{1, \ldots, \mathcal{L}\}$ **do**
    Randomly draw a permutation π from Per($\mathcal{C}$)
    $\mathbf{o}$ = Min-sum($\pi(\mathbf{o})$) /* Perform min-sum decoding with input $\pi(\mathbf{o})$, up to i_{max} iterations*/
    $\hat{\boldsymbol{u}}$ = Hard-Decision($\mathbf{o}$)
    $\mathbf{o} = \pi^{-1}(\mathbf{o})$, $\hat{\boldsymbol{u}} = \pi^{-1}(\hat{\boldsymbol{u}})$ /*Reorder the bits to its original (unpermuted) order*/.
    if $\hat{\boldsymbol{u}}\mathbf{H}^T = 0$ **then**
      $\mathbb{S} = \mathbb{S} \cup \{\hat{\boldsymbol{u}}\}$ /*Append a successfully decoded result to the candidate codeword set*/
      if $\lambda(\mathbf{r}, \hat{\boldsymbol{u}}) \leq G_T(\hat{\boldsymbol{u}}, d)$ **then**
        return $\hat{\boldsymbol{u}}$ /*End the decoding process immediately when the ML codeword is found.*/
      end if
      break /*End the j-loop if a codeword is found.*/
    end if
  end for
end for
if $|\mathbb{S}| \neq 0$ **then**
  return $\text{argmin}_{\hat{\boldsymbol{u}} \in \mathbb{S}} \lambda(\mathbf{r}, \hat{\boldsymbol{u}})$ /*Return the most likely codeword if there is no ML codeword identified.*/
else
  Declare a decoding failure.
end if

4. Simulation results and discussions

In this section, we will use the acronym "CEBP" to refer to the cyclically equivariant BP decoder in [17], and the acronyms "LMS" and "NLMS" to denote the layered min-sum and neural layered min-sum algorithms, respectively. It is worth mentioning that the CEBP can only be applied to cyclic redundant matrices. Similar to other neural BP-based decoders, e.g., [6,19], our proposed decoder is trained using 400,000 all-zero codewords corrupted by randomly generated Gaussian noise of SNR vary from 1 to 7 dB and $i_{max} = 5$ is set for all decoders. The optimizer used in training is RMSProp with learning rate of 0.01, and other parameters remain the same as PyTorch's default value. The batch size and number of epoch are set to 2000 and 10, respectively. For the test stage, randomly generated codewords and Gaussian noise are used for performance evaluation. All reported data points are obtained by simulation until no less than 100 error frames occur.

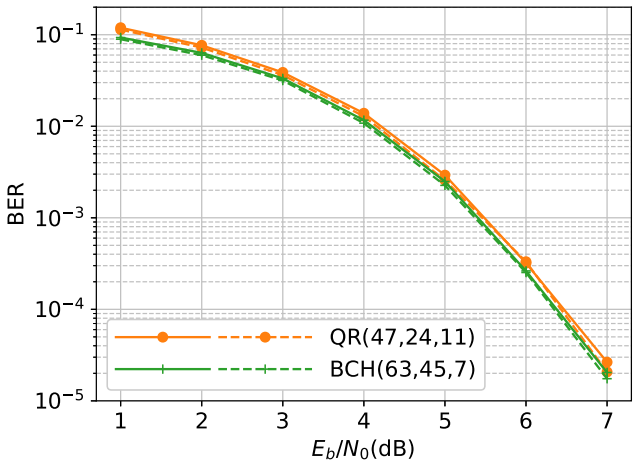


Fig. 2. The performance of NLMS over **H**, where dashed lines and solid lines correspond to the performance of edge-wise weighted NLMS and iteration-wise weighted NLMS, respectively.

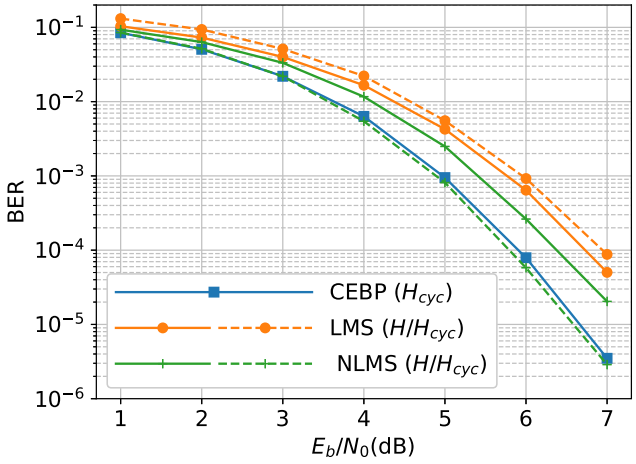


Fig. 3. Bit error rate (BER) performance of BCH(63,45,7) code.

4.1. Performance of NLMS decoding

Fig. 2 shows that our proposed decoder achieves similar performance with reduced number of weights compared to the decoder with edge-wise weights. Edge-wise weights are proposed to mitigate the effects of 4-cycles [6]. However, according to [23], instead of 4-cycles, small trapping sets are sometimes the main cause of performance degradation in high-density parity-check (HDPC) codes, and edge-wise weights cannot reduce the impact of trapping sets. Result in Fig. 2 also shows that edge-wise weight assignment may be unnecessary, and implies that neural BP-based decoders achieve better performance by properly attenuating the message instead of compensating short cycles. Because iteration-wise assignment cannot compensate for short cycles while can achieve the same performance as edge-wise assignment.

Simulation results of the (63,45,7) BCH code are shown in Fig. 3. The first row of its parity-check matrix (i.e. H_1) is (1, 1, 0,

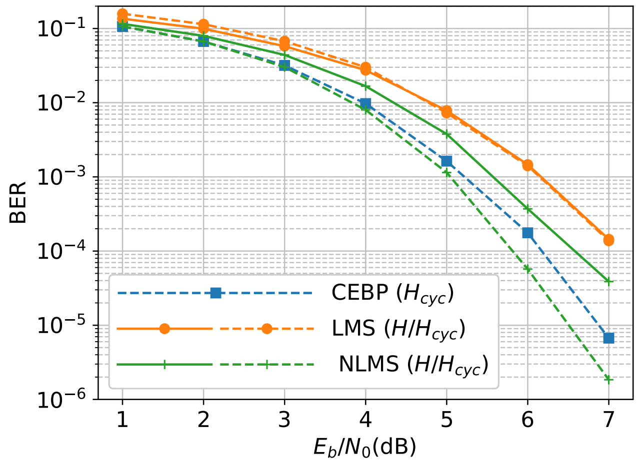


Fig. 4. BER performance of BCH(63,36,11) code.

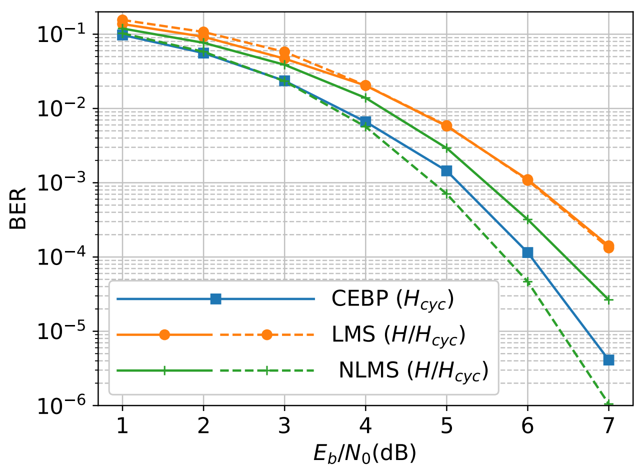


Fig. 5. BER performance of QR(47,24,11) code.

0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). It can be observed that the LMS has similar bit error rate (BER) performance over $\mathbf{H}$ and $\mathbf{H}_{cyc}$. Note that the NLMS decoder achieves a significant gain over the LMS counterpart in both cases of $\mathbf{H}$ and $\mathbf{H}_{cyc}$, and the improvement are 0.35 dB and 1.12 dB, respectively at BER $= 10^{-4}$. Moreover, it is found that the NLMS decoder with $\mathbf{H}_{cyc}$ significantly outperforms the one using the original matrix $\mathbf{H}$, and this phenomenon is also observed for codes of different lengths. Additionally, the performance gain of the NLMS decoder over the CEBP decoder is 0.06 dB at BER $= 4 \times 10^{-6}$.

The performance comparison of different decoders on the (63,36,11) BCH code is illustrated in Fig. 4. The H_1 of this code is (1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). It is noted that the NLMS decoder achieves a gain of 0.58 dB over the LMS decoder in the case of $\mathbf{H}$, and a gain of 1.20 dB in the case of $\mathbf{H}_{cyc}$ at BER $= 2 \times 10^{-4}$. Further, we found that the NLMS decoder outperforms the CEBP counterpart by 0.37 dB at BER $= 10^{-5}$.

The BER performance of different decoders on the (47,24,11) QR code is plotted in Fig. 5 . The H_1 of this code is (1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). It can be seen that the NLMS decoder achieves a gain of 0.64 dB over the LMS decoder using $\mathbf{H}$, and a gain of 1.18 dB in the case of $\mathbf{H}_{cyc}$ for BER $= 2 \times 10^{-4}$. We also observe that the NLMS decoder outperforms the CEBP decoder by 0.32 dB at BER $= 10^{-5}$.

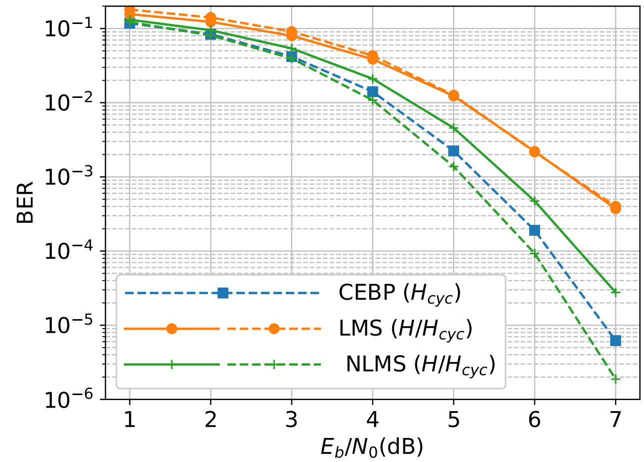


Fig. 6. BER performance of QR(71,36,11) code.

The BER curves of different decoders on the (71,36,11) QR code are depicted in Fig. 6 . The H_1 of this code is (1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). The NLMS decoder achieves a gain of 0.90 dB over the LMS decoder with $\mathbf{H}$ and a gain of 1.38 dB in the case of $\mathbf{H}_{cyc}$ at BER $= 4 \times 10^{-4}$. Moreover, it is noticed that the NLMS decoder outperforms the CEBP decoder by 0.29 dB at BER $= 10^{-5}$. The NLMS over $\mathbf{H}_{cyc}$ demonstrates superior performance, so we will use $\mathbf{H}_{cyc}$ throughout the remaining experiments.

4.2. Discussions on cyclic redundant matrices

The complexity of BP-based algorithms is proportional to the size of parity-check matrices and iterations. After applying the cyclic redundant matrix, each iteration is more complex than the decoder using the original matrix. For example, the original parity-check matrix $\mathbf{H}$ of the QR (47,24) code has the size of 23×47, and the cyclic redundant matrix $\mathbf{H}_{cyc}$ has the size of 47×47, which result in a 2.04 times complexity. As a result, we need to set the maximum iterations of the decoder using $\mathbf{H}$ to 2 times than that of the decoder using $\mathbf{H}_{cyc}$. For BCH (63, 45) code, this number should be 3.5 times.

From Fig. 7 we can see that the NLMS decoder based on $\mathbf{H}_{cyc}$ outperforms NOMS [14] based on $\mathbf{H}$ consistently. It is noted that $\mathbf{H}_{cyc}$ have much more short cycles than $\mathbf{H}$. For example, there are 1455 4-cycles and 57,130 6-cycles in the original $\mathbf{H}$ of the QR (47, 24) code, however, the $\mathbf{H}_{cyc}$ has 4371 4-cycles and 277,488 6-cycles. In contrast, 92 small trapping sets[2] were found in $\mathbf{H}$ but 0 was found in $\mathbf{H}_{cyc}$. This also explains why neural network decoders based on $\mathbf{H}_{cyc}$ can outperform that based on $\mathbf{H}$ at comparable complexity.

4.3. Discussions on the number of iterations

In previous works, e.g. [6,13,14,18,23], the neural network decoders are usually trained with a few iterations. One could argue that neural BP-based methods achieve better performance only by accelerating the convergence through message attenuation, consequently, these neural BP-based algorithms cannot

2 Here, "small trapping sets" refers to trapping sets satisfying $a+b < 8$, where a, b are the number of variable nodes and unsatisfied check nodes, respectively.

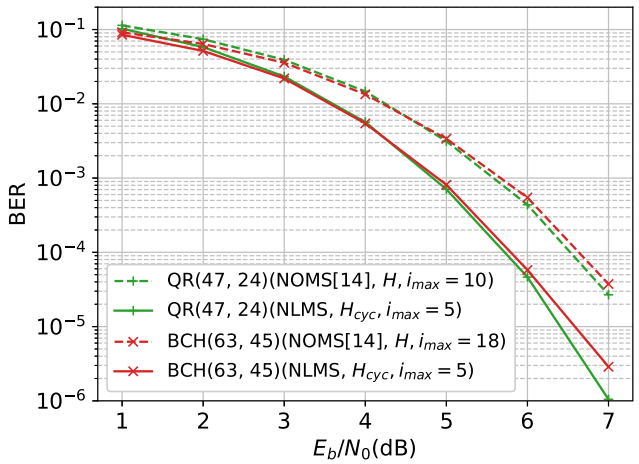


Fig. 7. BER performance of different decoders based on different matrices at a comparable complexity.

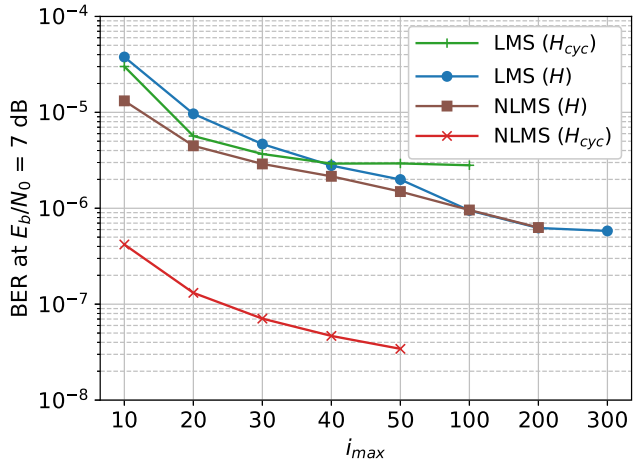


Fig. 8. BER performance of decoders for the QR (47, 24) with different maximum iterations.

outperform original BP-based algorithms when we increase the number of iterations. From Fig. 8 we can see that the performance gap between LMS and NLMS over **H** vanishes when the number of iterations reaches 100. This validates the argument we mentioned before. However, the NLMS over $\mathbf{H}_{cyc}$ consistently outperforms all the other decoders with a noticeable gain. We also noted that the LMS over $\mathbf{H}_{cyc}$ has higher BER than LMS over **H** when the number of iterations are large. This is probably because the $\mathbf{H}_{cyc}$ has more check nodes than the regular **H**, thus producing extreme extrinsic information magnitudes. But the NLMS can properly attenuate the extrinsic information to achieve a better error rate.

4.4. Performance of neural mRRD algorithm

The ML decoding performance of BCH codes in this section is cited from the database in [26]. The ML performance of QR and punctured RM codes is obtained by simulation, and the details are described in [27]. The "LD" refers to the list decoder with CEBP decoding of [17] and the term "NmRRD" refers to our proposed neural mRRD in **Algorithm** 1.

It can be seen from Fig. 9 that with the growing width $\mathcal{W}$ of decoders, both the NmRRD and LD can achieve near ML performance. For the NmRRD and LD with the best performance, the worst-case number of iterations of the LD is 5 times that of the NmRRD algorithm (i.e., $64\times20\times5 = 5\times(256\times5\times1)$). In addition,

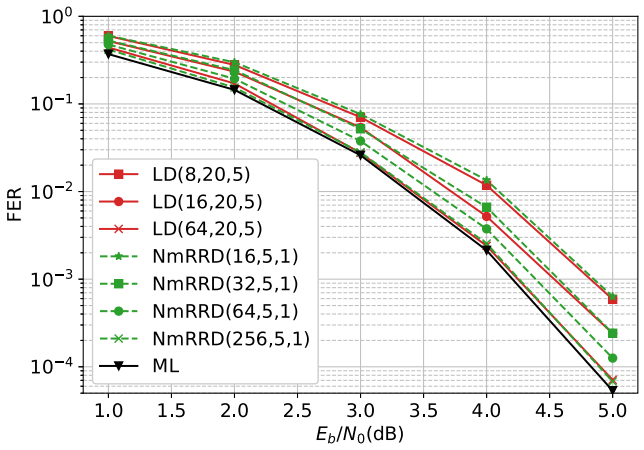


Fig. 9. FER performance of the BCH(63,45,7) code.

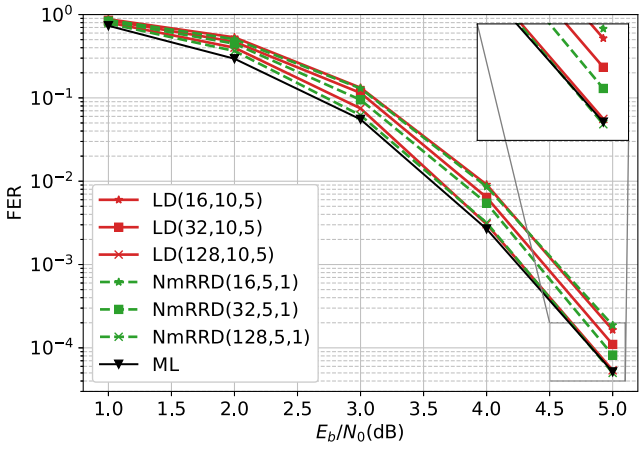


Fig. 10. FER performance of the punctured RM(127,99,7) code.

our mRRD(256,5,1) algorithm achieves a gain of 0.34 dB over the LD(16,20,5) with comparable number of iterations. Moreover, the NmRRD(16,5,1) and NmRRD(32,5,1) algorithms achieve similar performance as LD(8,20,5) and LD(16,20,5), respectively. But the number of iterations used in the NmRRD algorithm is reduced significantly by 90%.

Fig. 10 shows the FER performance of the NmRRD algorithm and LD algorithm with different sizes for the (127,99,7) punctured Reed–Muller (RM) code of which H_1 = (1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). Both the NmRRD and LD can achieve close ML performance. But our decoder can achieve similar FER performance with fewer iterations.

The FER performance of the NmRRD algorithm with different sizes for the (47,24,11) QR code is shown in Fig. 11. The performance of the LD on the QR code is absent since the affine permutations used in the LD cannot be applied directly to the QR code. Notably, one can see that the NmRRD(128,5,1) algorithm approaches the optimal ML decoding performance with a gap of 0.1 dB when FER is 1×10^{-5}.

5. Complexity analysis

We point out that both the proposed NLMS and NmRRD algorithms can be conducted in parallel. But for the sake of simplicity,

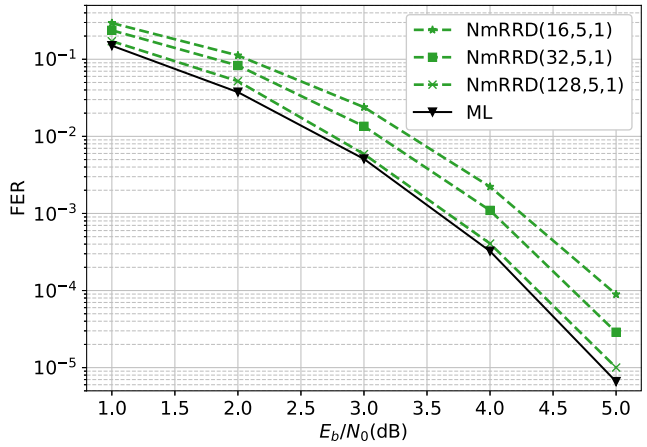


Fig. 11. FER performance of the QR(47,24,11) code.

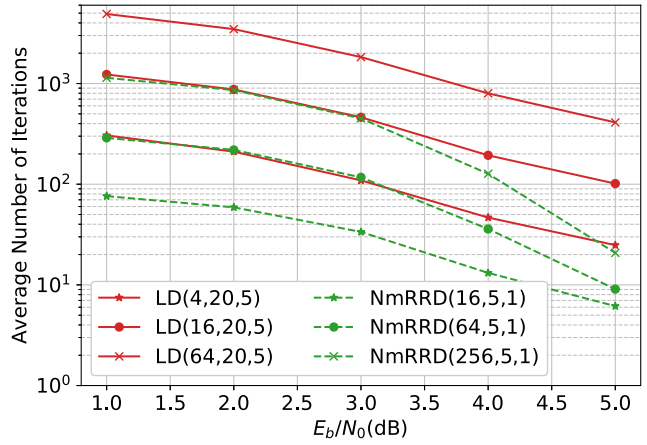


Fig. 12. Comparison of average iterations on the BCH(63,45,7) code.

the complexity analysis in Table 1 counts the total numbers of various operations, i.e. when all operations run serially. Given an $n \times n$ cyclic redundant matrix $\mathbf{H}_{cyc}$, let n_r be its row weight. Then the column weight of $\mathbf{H}_{cyc}$ is also n_r. For each iteration, the LD needs $n(n_r - 1)$ additions, $n(3n_r - 2)$ multiplications, n $\tanh(\cdot)$ operations and n $\tanh^{-1}(\cdot)$ operations. In contrast, each NmRRD iteration consists of $n(n_r - 1)$ additions, n multiplications, $n(n_r - 2)$ sign flips and n $\min(\cdot)$ operations. The complexities of the LD and the proposed NmRRD algorithm are evaluated by simulating 10,000 frames for each SNR and then calculating the average number of iterations, as shown in Fig. 12. Recall that the maximum number of iterations of the NmRRD algorithm is only 1/5 of that of the LD when similar near ML performance is attained. Now it can be noticed from Fig. 12 that the complexity reduction is further magnified with increasing E_b/N_0, e.g. the average number of iterations of NmRRD(256,5,1) is about 5% of that of LD(64,20,5) when $E_b/N_0 = 5$ dB, which could be attributed to the early stopping criterion in the proposed NmRRD algorithm.

6. Conclusion

This paper proposes a neural layered min-sum (NLMS) decoding algorithm to decode binary cyclic codes. Using the cyclic redundant matrix, we observed that the performance of the NLMS decoding could be greatly improved, while the performance improvement over the original parity-check matrix is limited. Moreover, the proposed NLMS decoder with the cyclic redundant matrix consistently outperforms the neural decoder proposed in [17], and it also has lower computational complexity. In order to approach the maximum likelihood (ML) decoding performance, we further combine the mRRD method with the proposed NLMS decoder. The resultant NmRRD algorithm achieves near ML performance for certain codes and reaches a significant reduction of computations compared to the list decoder of [17].

Table 1
The number of operations for different decoders on an (n, k, d) code.

Algorithm	$+$	$\times$	tanh	$\tanh^{-1}$	sign flips	min
LD	$n(n_r - 1)i$	$n(3n_r - 2)i$	$n \times i$	$n \times i$	0	0
NmRRD	$n(n_r - 1)i$	$n \times i$	0	0	$n(n_r - 2)i$	$n \times i$

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Source code available in https://github.com/Dies-Irae/Neural-Layered-Min-sum-Decoders-for-Cyclic-Codes.

Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant (No. U22A2026 and No.61771081), the Fundamental Research Funds for the Central Universities (China) under Grant cstc2019jcyjmsxmX0110. The work by Jianqing Liu was supported in part by National Science Foundation, China under grant ECCS-2312738.

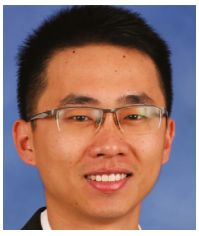
References

[1] K.-M. Cheung, F. Pollara, Phobos Lander Coding System: Software and Analysis, Telecommun and Data Acquisition Report, 1988.
[2] S.B. Wicker, Error Control Systems for Digital Communication and Storage, Prentice-Hall, Inc., USA, 1994.
[3] B. Honary, B. Hunt, M. Maundrell, Improving automatic link establishment through a new soft decision trellis decoder for the (24,12) Golay code, in: 1994 Sixth Int. Conference on HF Radio Systems and Techniques, 1994, pp. 182–185, http://dx.doi.org/10.1049/cp:19940489.
[4] M. Shirvanimoghaddam, M.S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li, S. Johnson, B. Vucetic, Short block-length codes for ultra-reliable low latency communications, IEEE Commun. Mag. 57 (2) (2019) 130–137, http://dx.doi.org/10.1109/MCOM.2018.1800181.
[5] I. Dimnik, Y. Be'ery, Improved random redundant iterative HDPC decoding, IEEE Trans. Commun. 57 (7) (2009) 1982–1985, http://dx.doi.org/10.1109/TCOMM.2009.07.070621.
[6] E. Nachmani, E. Marciano, L. Lugosch, W.J. Gross, D. Burshtein, Y. Be'ery, Deep learning methods for improved decoding of linear codes, IEEE J. Sel. Top. Signal Process. 12 (1) (2018) 119–131, http://dx.doi.org/10.1109/JSTSP.2017.2788405.
[7] H. Kim, Y. Jiang, R.B. Rana, S. Kannan, S. Oh, P. Viswanath, Communication algorithms via deep learning, in: Int. Conference on Learning Representations, 2018.
[8] Y. Zhang, H. Wu, M. Coates, On the design of channel coding autoencoders with arbitrary rates for ISI channels, IEEE Wirel. Commun. Lett. 11 (2) (2022) 426–430, http://dx.doi.org/10.1109/LWC.2021.3131848.
[9] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, P. Viswanath, LEARN codes: Inventing low-latency codes via recurrent neural networks, IEEE J. Sel. Areas Inf. Theory 1 (1) (2020) 207–216, http://dx.doi.org/10.1109/JSAIT.2020.2988577.

[10] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, P. Viswanath, Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels, in: NeurIPS, 2019, pp. 2754–2764.
[11] L. Huang, H. Zhang, R. Li, Y. Ge, J. Wang, AI coding: Learning to construct error correction codes, IEEE Trans. Commun. 68 (1) (2020) 26–39, http://dx.doi.org/10.1109/TCOMM.2019.2951403.
[12] P. Chen, L. Shi, Y. Fang, F.C.M. Lau, J. Cheng, Rate-diverse multiple access over Gaussian channels, IEEE Trans. Wireless Commun. (2023) 1, http://dx.doi.org/10.1109/TWC.2022.3233798.
[13] E. Nachmani, Y. Be'ery, D. Burshtein, Learning to decode linear codes using deep learning, in: 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2016, pp. 341–346, http://dx.doi.org/10.1109/ALLERTON.2016.7852251.
[14] L. Lugosch, W.J. Gross, Neural offset min-sum decoding, in: 2017 IEEE Int. Symp. on Info. Theory (ISIT), 2017, pp. 1361–1365, http://dx.doi.org/10.1109/ISIT.2017.8006751.
[15] K. Tian, C. Yue, C. She, Y. Li, B. Vucetic, A scalable graph neural network decoder for short block codes, 2022, arXiv:2211.06962.
[16] E. Nachmani, Y. Be'ery, Neural decoding with optimization of node activations, IEEE Commun. Lett. 26 (11) (2022) 2527–2531, http://dx.doi.org/10.1109/LCOMM.2022.3197974.
[17] X. Chen, M. Ye, Cyclically equivariant neural decoders for cyclic codes, in: Int. Conf. on Machine Learning (ICML), PMLR, 2021, pp. 1771–1780.
[18] E. Nachmani, L. Wolf, Hyper-graph-network decoders for block codes, in: Advances in Neural Information Proc. Systems, 32, Curran Associates, Inc., 2019.
[19] A. Buchberger, C. Häger, H.D. Pfister, L. Schmalen, A. Graell i Amat, Pruning and quantizing neural belief propagation decoders, IEEE J. on Sel. Areas in Comm. 39 (7) (2021) 1957–1966, http://dx.doi.org/10.1109/JSAC.2020.3041392.
[20] G. Li, X. Yu, Y. Luo, G. Wei, A bottom-up design methodology of neural min-sum decoders for LDPC codes, IET Commun. 17 (3) (2023) 377–386, http://dx.doi.org/10.1049/cmu2.12547.
[21] N. Shah, Y. Vasavada, Neural layered decoding of 5G LDPC codes, IEEE Commun. Lett. 25 (11) (2021) 3590–3593, http://dx.doi.org/10.1109/LCOMM.2021.3113610.
[22] D. Hocevar, A reduced complexity decoder architecture via layered decoding of LDPC codes, in: IEEE Workshop on Signal Proc. Systems, 2004, 2004, pp. 107–112, http://dx.doi.org/10.1109/SIPS.2004.1363033.
[23] M. Wang, Y. Li, R. Liu, H. Wu, Y. Hu, F.C.M. Lau, Decoding quadratic residue codes using deep neural networks, Electronics 11 (17) (2022) http://dx.doi.org/10.3390/electronics11172717.
[24] T. Kaneko, T. Nishijima, H. Inazumi, S. Hirasawa, An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder, IEEE Trans. Inform. Theory 40 (2) (1994) 320–327, http://dx.doi.org/10.1109/18.312155.
[25] Y. Li, P. Zhang, L. Wang, T.-K. Truong, Comments on "on decoding of the (89, 45, 17) quadratic residue code", IEEE Trans. Commun. 63 (2) (2015) 578–579, http://dx.doi.org/10.1109/TCOMM.2014.2386854.
[26] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, N. Wehn, Database of channel codes and ML simulation results, 2019, URL www.uni-kl.de/channel-codes.
[27] Y. Li, Q. Chen, H. Liu, T.-K. Truong, Performance and analysis of quadratic residue codes of lengths less than 100, 2014, http://dx.doi.org/10.48550/ARXIV.1408.5674, URL https://arxiv.org/abs/1408.5674.

Ming Wang is currently pursuing the Ph.D degree in Computer Science from North Carolina State University. He received the B.Eng. degree from Chongqing University of Posts and Telecommunications, Chongqing, China in 2020 and the M.Sc degree from Chongqing University. His research interests include channel coding, neural network and machine learning.

Yong Li received the B.Sc. degree in electronic and information engineering from the Chongqing University of Posts and Telecommunications (CQUPT), Chongqing, China, in 2003, and the M.S. and Ph.D. degrees in communication engineering from Xiamen University, Fujian, China, in 2006 and 2012, respectively. Since May 2018, he has been with the College of Computer Science, Chongqing University, where he is currently a Full Professor. He has published over 30 articles in peer-reviewed journals or conference proceedings, and held seven granted national patents and two U.S. patents. His primary research interests include channel coding, computer vision, and deep learning.



Jianqing Liu is currently an Assistant Professor at the Department of Computer Science at NC State University. He received a Ph.D. degree from The University of Florida in 2018 and a B.Eng. degree from the University of Electronic Science and Technology of China in 2013. His research interest is wireless communications and networking, security and privacy. He received the U.S. National Science Foundation Career Award in 2021. He is also the recipient of several best paper awards including the 2018 Best Journal Paper Award from IEEE Technical Committee on Green Communications & Computing (TCGCC)

Taolin Guo is currently an assistant researcher at the College of Computer Science in Chongqing University, China. He received the Ph.D. degree in Computer Science in 2020 from Southeast University. His research interests include security, privacy and social networks.

Huihui Wu received the B.Sc. degree in communication engineering from Southwest University for Nationalities, Chengdu, China, in 2011, and the M.S. degree in communication engineering from Xiamen University, Xiamen, China, in 2014. He received the Ph.D. degree in electrical and computer engineering from McMaster University, Hamilton, Canada, in 2018. From 2018 to 2019, he was a Postdoctoral Research Scientist with Columbia University, New York, USA. He is currently a Postdoctoral Researcher with McGill University, Montreal, Canada. His research interests include channel coding, joint source and channel coding, signal quantization, wireless communications, and deep learning.

Francis C.M. Lau (Fellow, IEEE/IET) received the B.Eng. degree (Hons.) in electrical and electronic engineering and the Ph.D. degree from King's College London, University of London, U.K. He is currently a Professor at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. He is the coauthor of two research monographs. He is also a co-holder of six U.S. patents. He has published more than 330 papers. His main research interests include channel coding, cooperative networks, wireless sensor networks, chaos-based digital communications, applications of complex-network theories, and wireless communications.