Improved Polar Decoder Based on Deep Learning

Weihong Xu[1,2], Zhizhen Wu[1,2], Yeong-Luh Ueng[3], Xiaohu You[2], and Chuan Zhang[1,2,*]
[1]Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
[2]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China
[3]Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan
Email: [2]{wh.xu, jasonwu, chzhang, xhyu}@seu.edu.cn, [3]ylueng@ee.nthu.edu.tw

***Abstract*—Deep learning recently shows strong competitiveness to improve polar code decoding. However, suffering from prohibitive training and computation complexity, the conventional deep neural network (DNN) is only possible for very short code length. In this paper, the main problems of deep learning in decoding are well solved. We first present the multiple scaled belief propagation (BP) algorithm, aiming at obtaining faster convergence and better performance. Based on this, deep neural network decoder (NND) with low complexity and latency, is proposed for any code length. The training only requires a small set of zero codewords. Besides, its computation complexity is close to the original BP. Experiment results show that the proposed $(64, 32)$ NND with 5 iterations achieves even lower bit error rate (BER) than the 30-iteration conventional BP and $(512, 256)$ NND also outperforms conventional BP decoder with same iterations. The hardware architecture of basic computation block is given and folding technique is also considered, saving about 50% hardware cost.**

***Index Terms*—Deep learning, polar code, deep neural network (DNN), belief propagation (BP).**

I. Introduction

Deep learning (DL) has attracted worldwide attentions in recent years due to its powerful capabilities to solve complex tasks. With the aid of deep learning, dramatic improvements are achieved in many fields, such as computer vision, game playing and bio-informatics. Moreover, human is defeated in GO chess and image recognition by state-of-the-art *Alpha GO* [1] and ResNet [2], which both adopt deep neural networks.

On the other side, polar codes have been regarded as an emerging breakthrough in channel coding because of their capacity-achieving property [3]. Now polar codes become one of the ideal forward error correction (FEC) candidates for the 5th generation (5G) wireless communication systems and storage systems.

Belief propagation (BP) decoding and successive cancellation (SC) decoding are two common decoding strategies for polar codes. Although SC decoding demonstrates its low complexity, it still suffers from high latency as well as limited throughput under long polar codes [4, 5]. Compared with SC decoding, BP decoding achieves much higher throughput and lower latency due to its parallelism. However, BP decoding also introduces a higher computational complexity. For the sake of reducing complexity and redundant computation, the min-sum (MS) approximation [6, 7] and early termination schemes [6, 8] are proposed recently. Nevertheless, they are still unable to obtain desirable performance within limited iterations.

In [9], a deep learning based decoder shows near optimal bit error rate (BER) performance by learning a large amount of codewords. But the exponential growing complexity both in training and running the neural network hinders practical application of long codewords. To overcome this obstacle, Cammere [10] replaces some sub-blocks of polar BP decoder with neural network aided components in [9] and shows good performance. Unfortunately, the decoding complexity remains prohibitive and hardware implementation with high throughput is hard to realize. Therefore, enormous training and running complexity are two main barriers preventing the application of deep learning in polar codes decoding.

This paper, for the first time, proposes a practical deep learning-aided model for any code length. Our contribution in this paper is threefold. First, we propose the multiple scaled BP algorithm. Then a novel deep neural network based polar decoder (NND) with low complexity and latency is presented. Second, we discuss training methodology in detail and prove the advantage of proposed NND in terms of complexity and latency. Third, the hardware architecture of basic computation block is given. And folding techniques [11] are adopted to optimize the basic computation block, reducing about 50% hardware cost.

The remainder of this paper is organized as below. Backgrounds of deep neural network and polar BP decoding are introduced in Section II. The existing challenges of deep learning in polar decoding are first analyzed in Section III. Then we present the multiple scaled BP decoding algorithm and the corresponding deep neural network decoder with low latency and high throughput is proposed. Section IV shows details of the proposed deep neural network decoder, its training procedure, and experiment results. Section V gives the hardware architecture as well as optimization of the proposed deep neural network decoder. Performance analysis is also shown. Section VI concludes this paper.

II. Preliminaries

A. Deep Neural Network

Deep neural network (DNN), also often called deep feed-forward neural network, is one of the quintessential deep learning models. A deep neural network model can be abstracted into a function f that maps the input $\mathbf{x}_0 \in \mathbb{R}^{N_0}$ to the output $\mathbf{y} \in \mathbb{R}^{N_L}$:

$$\mathbf{y} = f(\mathbf{x}_0; \theta), \tag{1}$$

978–1–5386–0446–5/17/$31.00 ©2017 IEEE

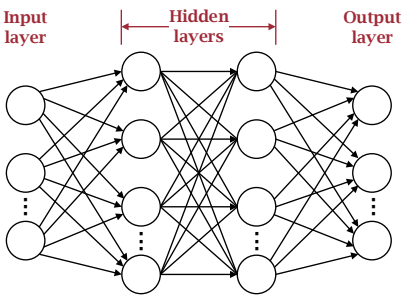


Fig. 1. Multi-layer architecture of deep neural network.

where θ denotes the parameters that result in the best function approximation of mapping the input data to desirable outputs.

In general, a DNN has a multi-layer structure, composing together many layers of function units (see Fig. 1). Between the input and output layers, there are multiple hidden layers. For an L-layer feed-forward neural network, the mapping function in the l-th layer with input $\mathbf{x}_{l-1}$ from $(l-1)$-th layer and output $\mathbf{x}_l$ propagated to the next layer can be defined as below:

$$\mathbf{x}_l = f^{(l)}(\mathbf{x}_{l-1}; \theta_l),\ l = 1, ..., L \tag{2}$$

where θ_l is the parameters of l-th layer, and $f^{(l)}(\mathbf{x}_{l-1}; \theta_l)$ is the mapping function in l-th layer.

To find the optimal set of all parameter $\theta = \{\theta_1, ..., \theta_l\}$, the mini-batch stochastic gradient descent (SGD) [12] is a popular method and will be introduced in Section III-C in detail.

B. Polar Codes

Polar codes exploit the channel polarization method to achieve the capacity of symmetric channel [3]. In the generation of an (N, K) polar code, K information bits and the other $(N-K)$ bits are first assigned to the reliable and unreliable positions of the N-bit message $\boldsymbol{u}^N$, respectively. Those bits in unreliable positions are referred as frozen bits and usually fixed to zeros. Then, the N-bit transmitted codeword $\boldsymbol{x}^N$ can be obtained through multiplying $\boldsymbol{u}^N$ with generator matrix $\boldsymbol{G}_N$ as following equation:

$$\boldsymbol{x}^N = \boldsymbol{u}^N \boldsymbol{G}_N = \boldsymbol{u}^N \boldsymbol{F}^{\otimes n} \boldsymbol{B}_N, \tag{3}$$

where $\boldsymbol{B}_N$ denotes the bit-reversal permutation matrix, $\boldsymbol{F}^{\otimes n}$ is the n-th *Kronecker* power of $\boldsymbol{F}$ and $n = \log_2 N$, $\boldsymbol{F} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$.

C. Belief Propagation Decoding

Belief propagation (BP) is a commonly used message passing algorithm for decoding. The BP algorithm can be conducted through performing iterative processing over the factor graph of any (N, K) polar code. In this case, the factor graph consists of $n = \log_2 N$ stages and $(n+1)N$ nodes in total.

The factor graph of $(8, 4)$ polar code is given in Fig. 2. Here, node (i, j) denotes the j-th input bit at the i-th stage. Two types of log likelihood ratios (LLRs), namely left-to-right message $R_{i,j}^{(t)}$ and right-to-left message $L_{i,j}^{(t)}$, are propagated iteratively

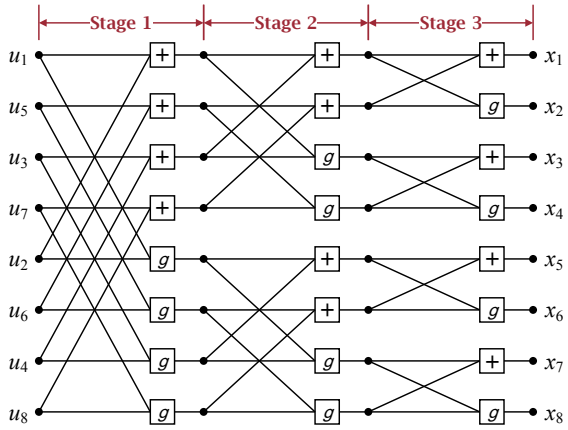


Fig. 2. Factor graph of polar codes with $N = 8$ [13].

over the factor graph, where t denotes the t-th iteration. The BP decoding for polar code is initialized as:

$$R_{1,j}^{(1)} = \begin{cases} 0, & \text{if } j \in \mathcal{A}, \\ +\infty, & \text{if } j \in \mathcal{A}^c, \end{cases} \quad L_{n+1,j}^{(1)} = \ln \frac{P(y_j|x_j=0)}{P(y_j|x_j=0)}. \tag{4}$$

It is noticed that $\mathcal{A}$ and $\mathcal{A}^c$ are the set of information bits and the set of frozen bits, respectively.

Then the iterative updating rules associated with $L_{i,j}^{(t)}$ and $R_{i,j}^{(t)}$ are represented by:

$$\begin{cases} L_{i,j}^{(t)} = g(L_{i+1,2j-1}^{(t)}, L_{i+1,2j}^{(t)} + R_{i,j+N/2}^{(t)}), \\ L_{i,j+N/2}^{(t)} = g(R_{i,j}^{(t)}, L_{i+1,2j-1}^{(t)}) + L_{i+1,2j}^{(t)}, \\ R_{i+1,2j-1}^{(t)} = g(R_{i,j}^{(t)}, L_{i+1,2j}^{(t)} + R_{i,j+N/2}^{(t)}), \\ R_{i+1,2j}^{(t)} = g(R_{i,j}^{(t)}, L_{i+1,2j-1}^{(t)}) + R_{i,j+N/2}^{(t)}, \end{cases} \tag{5}$$

where $g(x, y) = \ln \dfrac{(1+xy)}{x+y}$.

In order to reduce complexity, scaling min-sum, an approximation of above $g(x, y)$, is considered in Eq. (6).

$$g(x, y) \approx \alpha \cdot \text{sign}(x)\text{sign}(y) \min(|x|, |y|), \tag{6}$$

where α is the scaling parameter to offset the approximation error.

The estimation $\hat{\boldsymbol{u}}^N$ after T iterations is then judged by:

$$\hat{u}_j^N = \begin{cases} 0, & \text{if } L_{1,j}^T \geq 0, \\ 1, & \text{if } L_{1,j}^T < 0. \end{cases} \tag{7}$$

III. Proposed Deep Neural Network Decoder

A. Challenges of Deep Leaning in Polar Decoding

Many efforts have been made to decode polar code with less computation and better performance. The naive neural network decoder in [9] aims at recovering transmitted codeword after training. And [10] partly reduces the complexity. However, unlike limited categories in image recognition, the exponential nature of codeword means that there are 2^K categories for an (N, K) polar code. As the code length grows, training the neural network requires a huge collection of codeword dataset.

Besides, deep neural network usually stands for prohibitive computation demand, which is opposed to the high-speed and low-latency requirement of 5G.

The aforementioned methods do not take any features corresponding to conventional polar decoding strategies into consideration. If we are able to combine the structure of traditional polar decoding with deep learning techniques, it is likely to well balance complexity and performance.

B. Multiple Scaled BP Decoding

Pamuk first proposes min-sum approximation [7] to make BP decoding hardware-friendly. But it also introduces performance loss due to the approximation. As Eq. (6), Yuan points out that scaled min-sum [6] achieves similar performance of original BP algorithm with $\alpha = 0.9375$. The scaling parameter is unified for all iterations and all stages. It is supposed to obtain better results if we apply multiple α to different nodes during different iterations. Inspired by this, we propose the multiple scaled BP decoding algorithm:

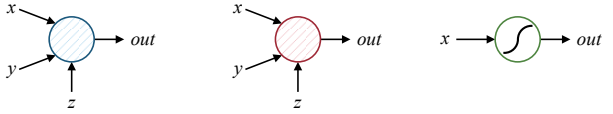
$$\begin{cases} L_{i,j}^{(t)} = \alpha_{i,j}^{(t)} \cdot g^{'}(L_{i+1,2j-1}^{(t)}, L_{i+1,2j}^{(t)} + R_{i,j+N/2}^{(t)}), \\ L_{i,j+N/2}^{(t)} = \alpha_{i,j+N/2}^{(t)} \cdot g^{'}(R_{i,j}^{(t)}, L_{i+1,2j-1}^{(t)}) + L_{i+1,2j}^{(t)}, \\ R_{i+1,2j-1}^{(t)} = \beta_{i+1,2j-1}^{(t)} \cdot g^{'}(R_{i,j}^{(t)}, L_{i+1,2j}^{(t-1)} + R_{i,j+N/2}^{(t)}), \\ R_{i+1,2j}^{(t)} = \beta_{i+1,2j}^{(t)} \cdot g^{'}(R_{i,j}^{(t)}, L_{i+1,2j-1}^{(t-1)}) + R_{i,j+N/2}^{(t)}, \end{cases} \tag{8}$$

where $\alpha_{i,j}^{(t)}$ and $\beta_{i,j}^{(t)}$ denotes the left-to-right and right-to-left scaling parameters of j-th node at i-th stage during t-th iteration, respectively. In this case, the function $g(x, y)$ in Eq. (6) is modified: $g^{'}(x, y) = \text{sign}(x)\text{sign}(y)\min(|x|, |y|)$.

By observing Eq. (8), our proposed multiple scaled BP is actually the general case of original min-sum [7] and scaled min-sum BP [6]. These algorithms are identical when $\alpha_{i,j}^{(t)} = \beta_{i,j}^{(t)} = 1$ or 0.9375. Therefore, once we properly select the parameters, it can be expected that the BER performance of multiple scaled BP is no worse than the above two algorithms.

C. Deep Neural Network Decoder

The scaling parameter α is generally selected through experience or experiment. There is no feasible methods to determine the optimal solution. However, with parameter scale in Eq. (8) increasing, it is difficult to find the best combination of $\alpha_{i,j}^{(t)}$ and $\beta_{i,j}^{(t)}$ by traditional approach.



(a) Neuron I (b) Neuron II (c) Sigmoid neuron

Fig. 3. Three types of neurons in deep neural network decoder.

Fortunately, deep neural network and deep learning techniques provide the powerful tools and enable us to optimize such a complex problem. This is resulted from the similarity between polar BP factor graph and deep neural network.

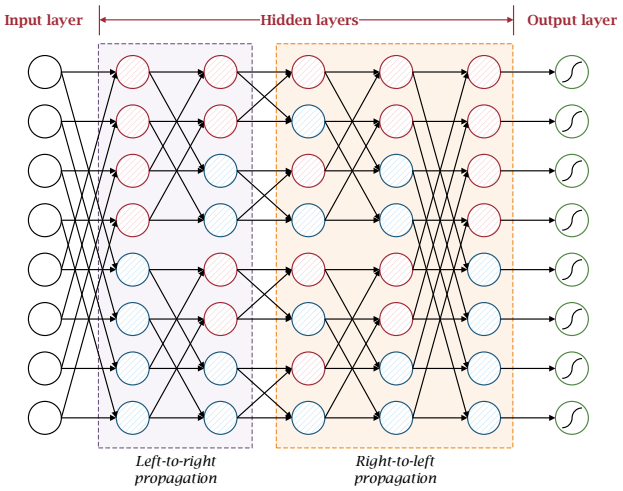


Fig. 4. One full BP iteration in neural network with $N = 8$.

More precisely, a deep neural network can be regarded as the unfolding structure of polar factor graph.

According to Eq.s (1), (2) and (8), the multiple scaled BP iterative functions in the l-th layer of an L-layer deep neural network can be rewritten as Eq. (9). Here we have $\mathbf{x}_{l-1} = \{\mathbf{x}_{l-1}^{'}, \mathbf{x}_{l-1}^{''}\}$ and $\mathbf{z} = \{\mathbf{z}^{'}, \mathbf{z}^{''}\}$,where $\mathbf{z}$ denotes the soft messages from the previous layer. To rescale the output into range $[0, 1]$, sigmoid activation function is used.

$$\begin{cases} f_{\text{I}}(\mathbf{x}_{l-1}; \theta_l)^{(l)} = \mathbf{W}_l^{\text{I}} \cdot g^{'}(\mathbf{x}_{l-1}^{'}, \mathbf{x}_{l-1}^{''} + \mathbf{z}^{'}), \\ f_{\text{II}}(\mathbf{x}_{l-1}; \theta_l)^{(l)} = \mathbf{W}_l^{\text{II}} \cdot g^{'}(\mathbf{x}_{l-1}^{'}, \mathbf{z}^{''}) + \mathbf{x}_{l-1}^{''}, \\ \boldsymbol{o} = o(\mathbf{x}_L) = \sigma(\mathbf{x}_{L-1}), \end{cases} \tag{9}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, $\boldsymbol{o}$ denotes the output vector and the weights to optimize becomes $\theta_l = \{\mathbf{W}_l^{\text{I}}, \mathbf{W}_l^{\text{II}}\}$, $\theta = \{\theta_1, ..., \theta_l\}$. Thereby, there are three types of neurons in our deep neural network decoder as Fig. 3 shows. The blue neuron, red neuron and green neuron correspond to f_{I}, f_{II} and $\sigma(x)$, respectively.

A simple example representing one full iteration in polar BP decoding is given in Fig. 4. For an (N, K) polar code, there are $n = \log_2(N)$ stages in the factor graph and each stage contains N nodes. After unfolding the original factor graph, every left-to-right and right-to-left propagation respectively correspond to $(n-1)$ and n layers in the feed-forward neural network structure and each layer consists of N neurons. The last hidden layer in right-to-left propagation calculates the output of leftmost nodes in original factor graph. Consequently, a full BP iteration is complete after input LLRs passing through $(2n-1)$ hidden layers. The last sigmoid output layer judges the estimation of received codewords. To increase the iterations in deep neural network, we only need to concatenate a certain amount of identical hidden layers in Fig. 4 between input layer and output layer. The proposed T-iteration deep neural network decoder is generally composed of $2(n-1)T+3$ layers.

To evaluate the decoding performance, cross entropy is adopted to express the expected loss of the neural network

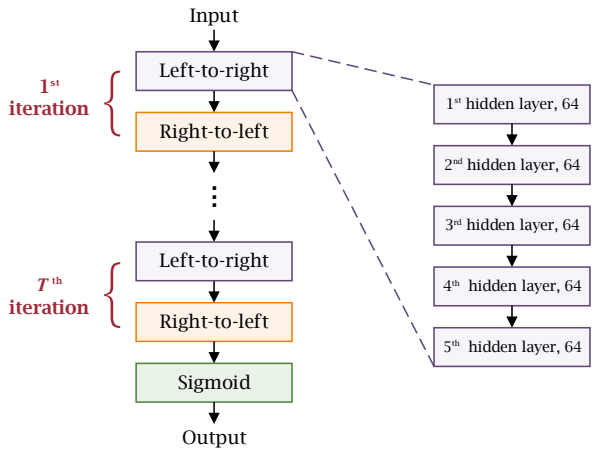


Fig. 5. Neural network architecture for $(64, 32)$ polar code with T iterations.

output $\boldsymbol{o}$ and the transmitted codeword $\boldsymbol{u}$:

$$L(\boldsymbol{u}, \boldsymbol{o}) = -\frac{1}{N}\sum_{i=1}^{N} u_i \log(o_i) + (1-u_i)\log(1-o_i). \quad (10)$$

The final goal for our model is to minimize the loss function in Eq. (10) and find the optimal parameters θ. One of the effective ways is mini-batch stochastic gradient descent (SGD):

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta_t \nabla L_M(\theta_t), \\ L_M(\theta_t) &= \frac{1}{M}\sum_{i \in \boldsymbol{M}} L(\boldsymbol{u}, \boldsymbol{o}), \end{aligned} \quad (11)$$

where $\boldsymbol{M}$ is mini-batch with size M randomly selected from training sets. And η_t denotes the learning rate at the t-th iteration, determining the size of updating step. The gradient is calculated through the back-propagation algorithm [12] efficiently. This can be done with the aid of the advanced DL libraries like Tensorflow [14].

IV. Experiments

A. Architecture of Deep Neural Network

To numerically demonstrate the advantages of proposed deep neural network decoder (NND), our model is tested with $(64, 32)$ polar code. We believe that our model will attain similar BER performance gain in the longer code length. In this case, the input LLRs will pass 5 hidden layers during one left-to-right or right-to-left propagation (see Fig. 5). And one full iteration corresponds to $5 \times 2 = 10$ hidden layers. As mentioned in Section III-C, the neural network for each BP iteration shares the same structure (see Fig. 4). Hence, a T-iteration decoder is realized by stacking T identical modules.

When implementing a deep NND, the other vital parameter is the network depth (or total iterations in BP). Deeper neural networks usually have greater capacities for decoding. However, in order to keep good balance between BER performance and complexity, the depth should be properly selected. There is no formulas or rules precisely measuring what the proper size of a specific neural network is. Hence, *greedy search algorithm* is applied to measure the optimal size of the network, which is explained as follows:

1) First, we start from a $(2(n-1)+3)$-layer neural network decoder where $n = \log_2 N$.
2) After the neural network is properly trained and converges, we feed the testing set to the trained network and obtain the testing BER performance.
3) Another $2(n-1)$ layers are stacked to the prior neural network and continue Step 2) until the BER plateaus.

B. Training Details

Our model is implemented on the advanced deep learning framework Tensorflow [14]. The gradient is calculated automatically. Training sets are all zero codewords after transmitting through an AWGN channel. The signal-to-noise ratios (SNRs) are ranging from 1dB to 4dB. We generate in total of 24000 codewords and the mini-batch size is 120 (30 codewords for each SNR). The adaptive method *Adam*[15] is applied with learning rate $=$ 0.001. To boost training, early stopping strategy is used. Weights initialization is a key part of neural network training. To avoid local optimization, we initialize our training parameters with all ones since the original BP factor graph has scaling parameters with all ones.

C. Results

We perform *Monte-Carlo* simulation to evaluate our deep NND. The testing set is generated by random binary messages (10000 frames for each SNR) after encoding, binary phase-shift keying (BPSK) modulation and transmitting through AWGN channel.

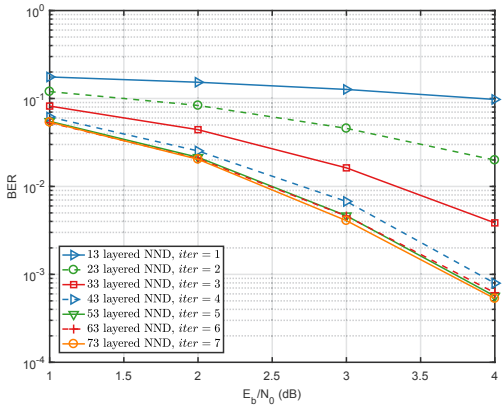


Fig. 6. BER performance $v.s.$ neural network depth for $(64, 32)$ polar code.

According to our *greedy search algorithm*, the proposed deep NND with varying depths is tested and the BER performance is presented in Fig. 6. The performance of 13-layer decoder is bad. But it improves very fast as the network gets deeper and becomes smooth around $iter = 5$. To this end, we select $T = 5$ as the desired iterations, which corresponds to a 53-layer neural network.

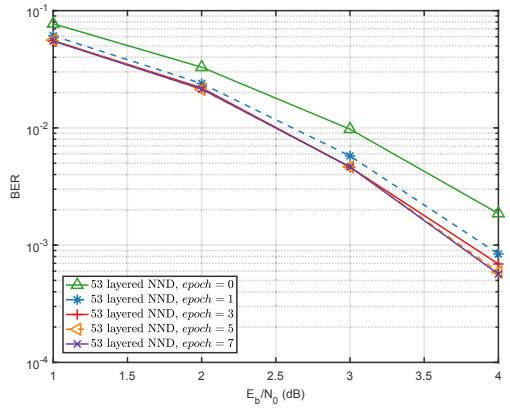


Fig. 7. Training results of $(64, 32)$ polar decoder.

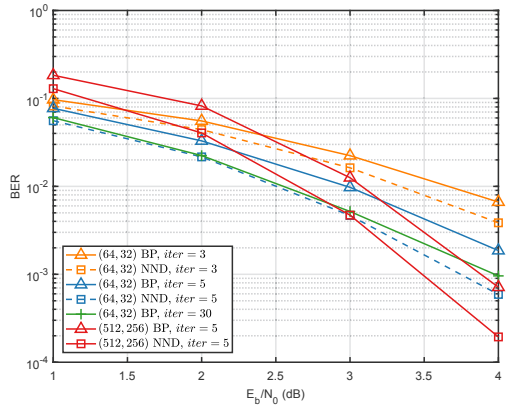


Fig. 8. BER comparison of BP and NND under with different code lengths.

Based on aforementioned results of *greedy search algorithm*, a 53-layer deep NND with 5 iterations is employed here. Fig. 7 shows the convergence situation during training. It can be observed that the BER curve becomes stable quickly around $epoch = 5$, indicating the convergence of training. Epoch denotes the times to expose all training data to the network.

Performance comparison between the proposed NND and the original BP with different code lengths is also conducted. From Fig. 8, slight improvement is gained in shallow neural network ($iter = 3$). Nevertheless, the proposed NNDs with 5 iterations under $(64, 32)$ and $(512, 256)$ codes, achieve much lower BER compared with conventional BP. The $(64, 32)$ NND outperforms 30-iteration BP. The advantage of NND tends to be more obvious in high SNR region. This is in agreement with our prediction that the proposed multiple scaled BP is superior to the min-sum BP.

The variation of scaling parameters $\mathbf{W}$ with respect to training epochs is given in Fig. 9. The histogram is close to standard normal distribution expect at 1. The weights scatter from epoch 1 to 3, suggesting that the decoder is learning multiple scaled min-sum BP. It is observed that the training process converges by the steady distribution after epoch 4.

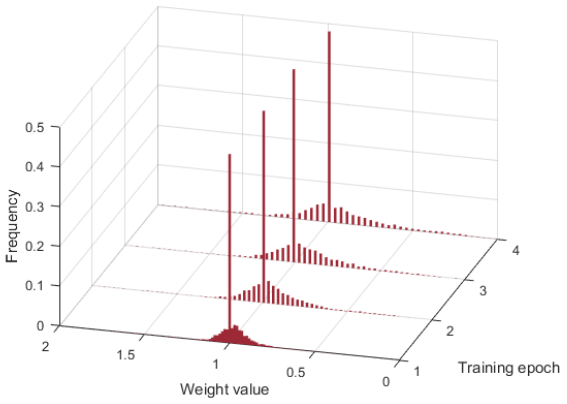
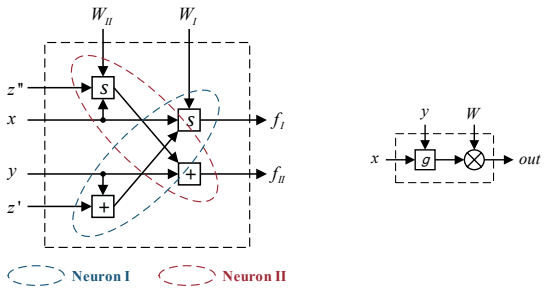


Fig. 9. Distribution of scaling parameters.

V. Hardware Implementation

A. Basic Computation Block

The proposed NND is hardware-friendly since it shares the identical factor graph with Fig. 2. Similar to the hardware architecture in [13], the modified *basic computation block* (BCB) of NND is illustrated in Fig. 10(a). The difference lies in the two extra inputs for scaling parameters. And scaling block s replaces the original g block. And one multiplier in s (see Fig. 10(b)) is required to compute the scaled output LLR.



(a) Modified BCB for NND. (b) Scaling block s.

Fig. 10. Basic blocks of NND.

B. Folded BCB

The proposed NND reduces decoding latency and improves throughput, but the scaling block to a certain extent increases the hardware cost compared to original BP. One modified BCB in Fig. 10(a) consists of 2 scaling blocks and 2 adders. These modules can be reduced through multiplexing them. We apply folding techniques [11] to save hardware area. The data flow graph (DFG) of BCB is redrawn as Fig. 11.

We select 2 as the folding factor and the following folding set is considered:

$$S = \{s_0, s_1\}, \ A = \{a_0, a_1\}. \tag{12}$$

After applying folding techniques, the folded architecture of modified BCB is illustrated as Fig. 12. Each folded BCB

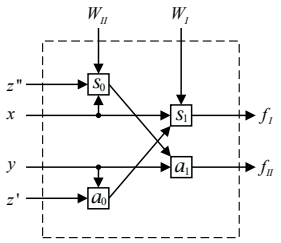


Fig. 11. Data flow graph (DFG) of BCB.

only requires one scaling block and one adder, saving about 50% hardware cost compared to original modified BCB. The latency of one folded BCB consequently doubles.

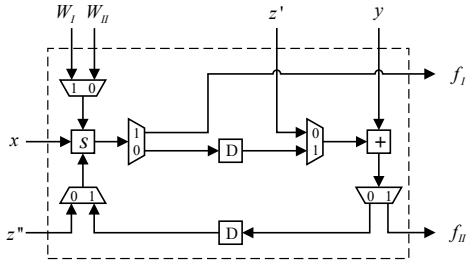


Fig. 12. Folded BCB for NND.

C. Comparison

In order to conduct a fair comparison, we consider the fully parallel BP decoder for (N, K) polar code. Hardware cost comparison is summarized in Table I.

TABLE I
HARDWARE COSTS OF FOLDED NND AND MIN-SUM BP.

Hardware costs	Proposed NND	Min-sum BP
BCB	$N(\log_2 N - 1)$	$N(\log_2 N - 1)$
Adder	$N(\log_2 N - 1)$	$2N(\log_2 N - 1)$
***g* block**	$N(\log_2 N - 1)$	$2N(\log_2 N - 1)$
Multiplier	$N(\log_2 N - 1)$	0

The proposed NND using folded BCB consumes 50% adders and g blocks than the original min-sum BP. And the BER improvement of NND is at the expense of $N(\log_2 N - 1)$ multipliers. Decoding latency is one key metric to evaluate the performance of a polar decoder. Assuming that processing time t_p for each stage of NND and BP is the same, the deep NND in Section IV with folded BCB has a constant latency:

$$T_{\text{latency}}^{\text{NND}} = [4(\log_2 N - 1)T + 1]t_p, \tag{13}$$

where T in this paper equals to 5.

While for the conventional BP decoder, the latency is:

$$T_{\text{latency}}^{\text{BP}} = [2(\log_2 N - 1)I + 1]t_p, \tag{14}$$

where I is the total iterations to decode a frame codeword.

Typically, the conventional min-sum BP requires 30 to 40 iterations to achieve ideal BER performance, especially in low SNR range. However, the proposed NND is able to obtain promising or even lower BER with only 5 iterations, resulting much lower latency and complexity than the min-sum BP one.

VI. CONCLUSION

In this paper, a novel deep neural network based polar decoder is proposed, which outperforms the conventional BP decoder within fewer iterations. It should be noted that this design methodology is general and can be applied to other BP-based decoders. Future work will focus on further optimization for neural network architecture and hardware performance.

ACKNOWLEDGMENT

This work is partially supported by NSFC under grant 61501116, International Science & Technology Cooperation Program of China under grant 2014DFA11640, Huawei HIRP Flagship under grant YB201504, Jiangsu Provincial NSF under grant BK20140636, ICRI for MNC, the Fundamental Research Funds for the Central Universities, State Key Laboratory of ASIC & System under grant 2016KF007, and the Project Sponsored by the SRF for the Returned Overseas Chinese Scholars of MoE.

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of GO with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[3] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul 2009.

[4] C. Zhang and K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2429–2441, 2013.

[5] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency SC polar decoder architectures," in *Proc. IEEE International Conference on Communications (ICC)*, June 2012, pp. 3471–3475.

[6] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6496–6506, Dec 2014.

[7] A. Pamuk, "An FPGA implementation architecture for decoding of polar codes," in *International Symposium on Wireless Communication Systems (ISWCS)*, Nov 2011.

[8] Y. Ren, C. Zhang, X. Liu, and X. You, "Efficient early termination schemes for belief-propagation decoding of polar codes," in *IEEE International Conference on ASIC (ASICON)*, Nov 2015.

[9] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," *arXiv preprint arXiv:1701.07738*, 2017.

[10] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," *arXiv preprint arXiv:1702.06901*, 2017.

[11] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley & Sons Inc, 1999.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] J. Yang, C. Zhang, H. Zhou, and X. You, "Pipelined belief propagation polar decoders," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016.

[14] M. Abadi, A. Agarwal, P. Barham *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.