Polar Decoding on Sparse Graphs with Deep Learning

Weihong Xu[1,2], Xiaohu You[2], Chuan Zhang[1,2] and Yair Be'ery[3]
[1]Lab of Efficient Architecture for Digital Communication and Signal Processing (LEADS)
[2]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China
[3]School of Electrical Engineering, Tel-Aviv University, Israel
Email: [2]{wh.xu, xhyu, chzhang}@seu.edu.cn, [3]ybeery@eng.tau.ac.il

***Abstract*—In this paper, we present a sparse neural network decoder (SNND) of polar codes based on belief propagation (BP) and deep learning. At first, the conventional factor graph of polar BP decoding is converted to the bipartite Tanner graph similar to low-density parity-check (LDPC) codes. Then the Tanner graph is unfolded and translated into the graphical representation of deep neural network (DNN). The complex sum-product algorithm (SPA) is modified to min-sum (MS) approximation with low complexity. We dramatically reduce the number of weight by using single weight to parameterize the networks. Optimized by the training techniques of deep learning, proposed SNND achieves comparative decoding performance of SPA and obtains about 0.5 dB gain over MS decoding on $(128, 64)$ and $(256, 128)$ codes. Moreover, 60% complexity reduction is achieved and the decoding latency is significantly lower than the conventional polar BP.**

***Index Terms*—Polar codes, belief propagation, deep learning, neural networks, sparse graphs.**

I. Introduction

Deep neural network (DNN) and deep learning techniques show promising performance in vast variety of tasks. In quantum information, DNN is utilized to decode stabilizer code [1] through encoding the probability distribution of errors. The authors in [2] discuss the possibility of applying DNN to channel equalization and decoding. Recurrent neural network (RNN) is adopted to detect data sequences [3] in communication systems.

On the other side, polar codes [4] are regarded as a prominent breakthrough in channel coding because of their capacity-achieving property. Now polar codes have been selected as the error-correcting codes of the enhanced mobile broadband (eMBB) control channels for the 5th generation (5G) wireless communication systems. With the advanced deep learning libraries and high performance hardware, many efforts have been made to develop a neural network decoder (NND) that can adaptively decode polar codes under different channel conditions. The authors in [5] exploit naive dense neural network to decode very short polar codes. It shows that NND trained by all possible codewords leads to near maximum a posteriori (MAP) performance. But the complexity is prohibitive due to the exponential nature of binary codewords. To alleviate the enormous complexity of long polar codes, [6] partitions the polar encoding graph into small blocks and train them individually. Although the degradation of partitioning is negligible, the overall decoding complexity is still high. To overcome these issues, in [7], trainable weights are assigned to the edges of belief propagation (BP) factor graph and then the iterative BP decoding is converted into DNN. The method requires much lower complexity and less parameters compared to [5, 6], which is feasible for long polar codes. However, the decoding latency is long since the depth of NND is determined by iteration number and code length.

In this work, we propose a sparse neural network decoder (SNND) for polar codes with high parallelism, low latency and low complexity. Inspired by [8], our SNND is constructed from the bipartite Tanner graph of polar codes in [9]. The sum-product algorithm (SPA) is replaced by min-sum (MS) approximation to reduce complexity. After the network is trained by deep learning techniques, SNND achieves the equal bit error rate (BER) performance with SPA decoding. Moreover, the decoding latency is about $1/\log_2 N$ of the conventional polar BP [10] due to the fully parallel structure.

The remainder of this paper is organized as below. Polar codes and BP decoding are briefly introduced in Section II. Section III describes how to construct the sparse trellis of SNND. Then the corresponding decoding process and model training methodology are given in detail. The experiment results in Section IV demonstrate the improvements of proposed SNND over various code lengths. The latency and complexity analysis is also given. Section V concludes this paper.

II. Preliminaries

A. Polar Codes

Polar codes have proven to be capable of achieving the capacity of symmetric channel [4]. The encoder of an (N, K) polar code assigns K information bits and the other $(N-K)$ bits to the reliable and unreliable positions of the N-bit codeword $\mathbf{u}^N$, respectively. Those bits in unreliable positions are referred as frozen bits and usually fixed to zeros. Then, the N-bit transmitted codeword $\mathbf{x}^N$ can be obtained according to $\mathbf{x}^N = \mathbf{u}^N\mathbf{G}_N$, where $\mathbf{G}_N$ is the generator matrix and satisfies $\mathbf{G}_N = \mathbf{F}^{\otimes n}$. Note that $\mathbf{F}^{\otimes n}$ is the n-th *Kronecker* power of $\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $n = \log_2 N$.

B. Belief Propagation Decoding

BP is one of the commonly used message passing algorithms for polar decoding. The BP algorithm decodes polar

codes through iteratively processing the log-likelihood ratios (LLRs) over the factor graph of any (N, K) polar code. Unlike the fully parallel Tanner graph of LDPC codes, the factor graph of polar decoding is based BP decoder for Reed-Muller (RM) codes. In this case, the factor graph consists of $n = \log_2 N$ stages and $(n+1)N$ nodes in total. Fig. 1 illustrates the factor graph of $(8, 4)$ polar code.

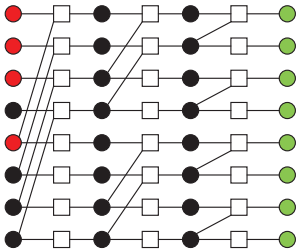
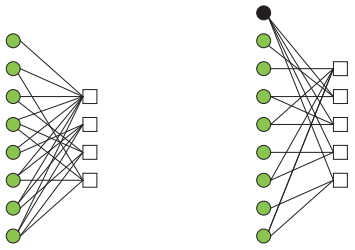


Fig. 1. Factor graph of (8,4) polar code with $\mathcal{A} = \{4, 6, 7, 8\}$ [9].

C. LDPC-like Polar Decoder

The polar BP decoder is generally constructed based the generator matrix $\mathbf{G}_N$, which has a similar trellis structure with its encoding factor graph. However, this causes inefficiencies for polar decoding since the number of stages is determined by the code length. Moreover, the multiple-stage architecture of polar decoder results in longer latency compared with the fully parallel scheduling of LDPC-like BP decoding.



(a) Dense (b) Sparse

Fig. 2. LDPC-like Tanner graphs [9] for (8,4) polar code with $\mathcal{A} = \{4, 6, 7, 8\}$.

To overcome the aforementioned problems, the parity-check matrix $\mathbf{H}$ of polar codes can be constructed from the corresponding generator matrix $\mathbf{G}_N$ in [11]. The conventional polar BP factor graph is then converted to the LDPC-like bipartite graph (see Fig. 2(a)) consisting of variable nodes (VNs) and check nodes (CNs). But the dense graph representation involved with many circles has demonstrated to show poor performance over additive white Gaussian noise (AWGN) channel [9]. Pruning methods for polar factor graph are consequently proposed in [9] to perform efficient polar decoding with LDPC-like manner. The sparse graph after using pruning techniques is shown as Fig. 2(b). For more details, we refer the readers to [9, 11].

III. Proposed Sparse Neural Network Decoder

A. Trellis Construction of Sparse Neural Network Decoder

The trellis of proposed SNND is constructed based on the sparse polar Tanner graph in [9]. More specifically, proposed SNND is a deep feed-forward neural network similar to the structure of [8]. The nodes of each hidden layer represent corresponding edges in the Tanner graph.

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{1}$$

The trellis construction of (8,4) polar SNND is given as an example. The conventional factor graph associated with generator matrix $\mathbf{G}_8 = \mathbf{F}^{\otimes 3}$ is first converted into the LDPC-like Tanner graph (see Fig. 2(a)) consisting of VNs and CNs [11]. Then we use the pruning techniques of [9] to reduce the number of edges, converting the dense graph into a sparse Tanner graph (see Fig. 2(b)). The resulting parity-check matrix $\mathbf{H}$ is shown in Eq. (1). Note that the sparse Tanner graph is slightly different from LDPC codes since a portion of edges from VNs to CNs are not removed [9] (*black VN* in Fig. 2(b)).

Next, the bipartite sparse Tanner graph is unfolded and converted into the feed-forward neural network in Fig. 3. Assume that we have an (N, K) polar code on sparse Tanner graph with total E edges, N_v VNs, and T iterations in the sparse Tanner graph. The associated SNND has $2T$ hidden layers. For the input layer with N_v nodes, the initial LLRs of received channel output are fed into the last N nodes. The number of nodes in each hidden layer equals to the edges E and each hidden node denotes the soft message propagated over corresponding edge. The final N_v outputs are activated by the sigmoid function.

B. Decoding Process

Let $\mathbf{x} = (x_1, ..., x_N)$ be the transmitted codeword with systematic encoding [12] and $\mathbf{y} = (y_1, ..., y_N)$ be the received channel output. The input size of SNND is slightly larger than N since part of VNs are not removed. The initial LLR of the v-th node in input layer is computed as the following equation:

$$L_v = \begin{cases} 0, & 1 \le v \le N_v - N, \\ \log \dfrac{P(x_j = 0|y_j)}{P(x_j = 1|y_j)}, & N_v - N + 1 \le v \le N_v, \end{cases} \tag{2}$$

where we have $j = v - (N_v - N)$.

The standard SPA can be used to construct polar codes over Tanner graphs as [8, 13]. But the computational complexity of SPA is prohibitive due to the hyperbolic trigonometric function and multiplication. [14] demonstrates that NND constructed by MS decoding can also achieve promising performance compared with SPA. Therefore we use the simplified MS decoding to define the two types of basic neurons in SNND see Fig. 3. Each neuron represents the associated edge in Tanner

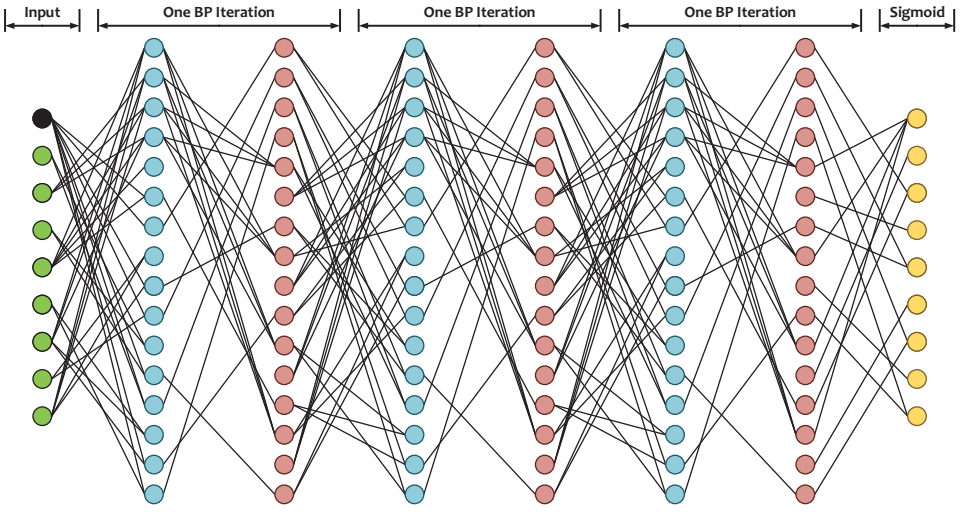


Fig. 3. Sparse neural network decoder (SNND) for (8,4) polar code with 6 hidden layers.

graph. The odd layer i only contains neurons without any parameters. The updating function is the MS approximation:

$$x_{i,e=(c,v)} = \prod_{e'=(v',c),v'\neq v} \text{sign}(x_{i-1,e'}) \cdot \min(|x_{i-1,e'}|), \quad (3)$$

where $e' = (v', c)$ denotes the set of VNs v' connected to CNs c.

The even hidden layer i only contains neurons that assign weights to incoming messages as follows:

$$x_{i,e=(v,c)} = L_v + \sum_{e'=(c',v),c'\neq c} w_{i,e,e'} x_{i-1,e'}. \quad (4)$$

The output layer squashes the final weighted soft messages to the range $[0, 1]$ as follows:

$$o_v = \sigma(L_v + \sum_{e'=(c',v)} w_{2L+1,v,e'} x_{2L,e'}), \quad (5)$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function. Note that the sigmoid function is only applied to the output layer during training phase. For simplicity, the feed-forward SNND is defined as SNND-FF.

C. Optimizing with Single Weight

The decoding complexity of SNND is significantly reduced compared to the original SPA. However, the required number of weights is still large. For the RNNs in [14], the weights of edges are shared within each BP iteration. Besides, the RNN structure is easier to optimize compared to the feed-forward counterparts. There is still some redundancy for the RNN structure. We further reduce the required number of weights to just one as follows:

$$x_{i,e=(v,c)} = L_v + \sum_{e'=(c',v),c'\neq c} w' x_{i-1,e'}, \quad (6)$$

where w' denotes the unified weight for all edges from CNs to VNs. w' is also applied to the final output in Eq. (5).

The optimization is easier and the optimal parameter w^* is given by w that results in the minimum loss:

$$w^* = \arg\min_{w'} \mathcal{L}(\boldsymbol{x}, \boldsymbol{o}). \quad (7)$$

D. Training of Sparse Neural Network Decoder

The cross entropy function is adopted to express the evaluate the loss between neural network output $\mathbf{o}$ and the transmitted codeword $\mathbf{x}$:

$$\mathcal{L}(\mathbf{x}, \mathbf{o}) = -\frac{1}{N} \sum_{i=N'}^{N_v} x_i \log(o_i) + (1 - x_i) \log(1 - o_i), \quad (8)$$

where o_i, x_i denote the i-th bit of SNND outputs and the i-th bit of transmitted codeword, respectively. The last N bits are calculated and $N' = N_v - N + 1$.

The parameter space of SNND is determined by the total edges in corresponding sparse Tanner graph and the iteration number. Hence, the optimization space grows larger when the code length and the iteration increase. A good parameter initialization can boost the convergence of training. [15] suggests to initialize the parameters with a normal distribution. But the standard normal distribution is unable to guarantee a quick convergence. We initialize the parameters of feed-forward SNND to a normal distribution with mean $\mu = 1$ and a small variance σ in the experiment while the SNND with single weight is initialized to one.

IV. Experiment

A. Setup

The SNND is implemented on deep learning library *PyTorch*. We use mini-batch stochastic gradient descent (SGD) with Adam [16] algorithm to optimize the neural network. The learning rate Lr is set to 0.001. AWGN channel and binary phase-shift keying (BPSK) modulation with SNR range 1 to 4 are considered. As in [8], the training set consists of all zero codeword and the mini-batch size is 120 (30 samples per

SNR). The parameters are initialized with normal distribution $\mathcal{N} \sim (\mu = 1, \sigma = 0.1)$. Zero-value messages in the SNND will make the CN-to-VN messages in Eq. (3) to be zero, which hinders the message propagation. To avoid this issue, the result of sign operation for a zero value is defined as 1.

B. Results

We train two types of SNND: SNND-FF and SNND with single weight. Both of them are unfolded to 10 iterations, corresponding to 20-layered neural networks. Each network is trained for 600 epochs. Fig. 4 illustrates the trend of trained unified weight w' on $(128, 64)$ and $(256, 128)$ polar codes. The trained optimal w^* for $(128, 64)$ code finally converges to 0.83 while the value of w^* for $(256, 128)$ code is closed to 0.82.

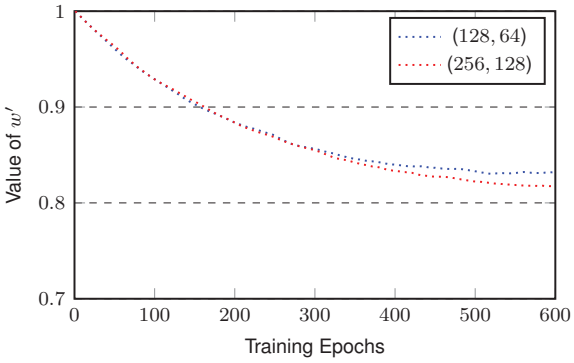


Fig. 4. Evolution of trained weight w' of SNND.

The BER performance is evaluated with four types of decoding schemes: 1) SPA on sparse Tanner graph [9], 2) MS algorithm on sparse Tanner graph, 3) Proposed SNND-FF, 4) Proposed SNND with single weight. Fig. 5 illustrates the BER results for two trained SNNDs on $(128, 64)$ polar code. The SNND-FF equivalent to 10 iterations achieves almost the same performance with SPA and has an improvement of about 0.1 to 0.5 dB over MS decoding. The gap between SNND-FF and SNND with single weight is negligible. Fig. 6 shows the BER comparison on $(256, 128)$ polar code. The SNNDs have about 0.15 dB gain over MS decoding and have less than 0.1 dB performance degradation in high SNR region compared with SPA.

We also compare the SNND with other decoding algorithms. The scaled min-sum (SMS) and neural network decoder (NND) in [7] are considered. The scaling factor of SMS equals to 0.9375, which is suggested in [17]. The NND is trained by unfolding to 10 iterations and tested with 50 iterations. After increasing the iteration number, the SNND with single weight w^* can obtain better performance. Fig. 7 shows the performance comparison for various decoding schemes with 50 iterations. The SNND with trained w^* achieves comparative performance with SPA and outperforms SMS and MS by about 0.1 dB and 0.4 dB, respectively. Due to pruning of some connections, the SNND has 0.1 dB degradation compared with polar NND in [7]. The similar results can also be observed on $(256, 128)$ polar code in Fig. 8.

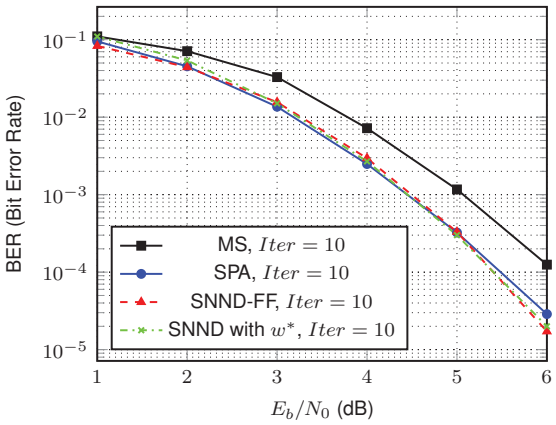


Fig. 5. BER comparison of trained SNNDs and different decoding schemes on $(128, 64)$ polar code with 10 iterations.

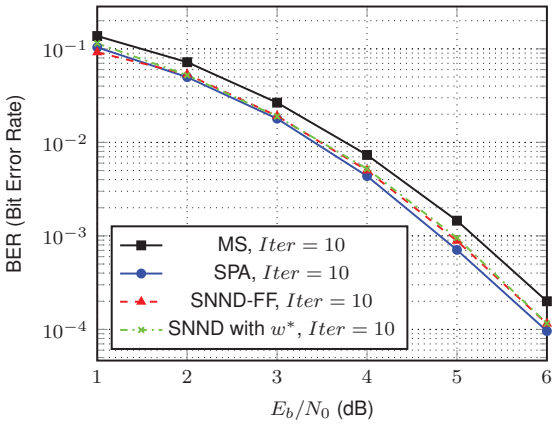


Fig. 6. BER comparison of trained SNNDs and different decoding schemes on $(256, 128)$ polar code with 10 iterations.

C. Complexity and Latency Analysis

The latency and complexity of proposed SNND are both reduced compared to NND in [7] and the original polar BP decoding with SMS [17]. The original polar BP [10] will consecutively activate $2\log_2 N$ stages during left-to-right and right-to-left propagations, resulting a latency of $2\log_2 N$ time steps for each iteration. Besides, N multiplications, N additions, and N comparisons are required for each stage. Hence, the total complexity of one iteration is $\mathcal{O}(2N\log_2 N)$.

The complexity of SNND with single weight is determined by corresponding $\mathbf{H}$ matrix. Each CN with d_c incoming messages requires $2d_c$ comparisons to find the minimum and 2nd minimum value. 2 multiplications are needed to compute the outgoing messages. Each VN with d_v incoming messages requires d_v additions. Note that the sign operation of CNs is omitted since its complexity is very low. SNND implements a LDPC-like flooding pattern with high parallelism. The latency for each iteration equals to 2, which is independent with code length. Hence, the latency reduction is $\log_2 N$ compared with the NND [7] and original BP. Fig. 9 gives the number of three types of operations (addition, multiplication and comparison)

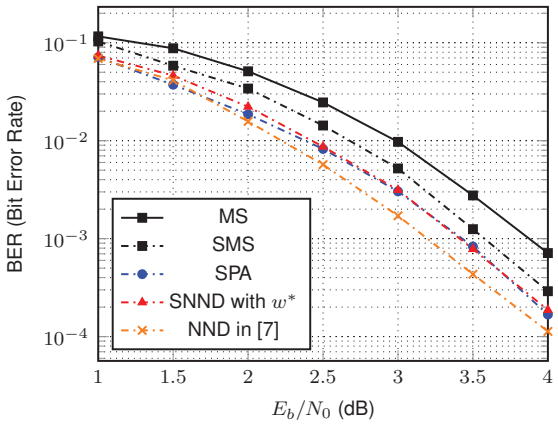


Fig. 7. BER comparison for various decoding schemes on $(128, 64)$ polar code with 50 iterations.

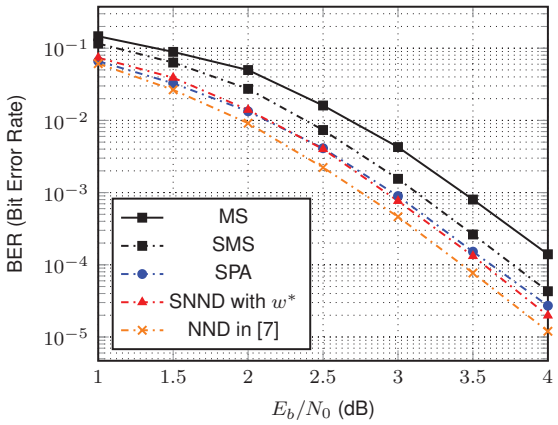


Fig. 8. BER comparison for various decoding schemes on $(256, 128)$ polar code with 50 iterations.

for proposed SNND with single weight and NND in [7]. The SNND can reduce about 60% operations on the two mentioned code lengths.

V. Conclusion

In this work, we propose a fully parallel neural network decoder for polar codes. The SNND is constructed from the sparse Tanner graph of polar codes [9]. Then the weights of SNND are dramatically reduced to just one by using single parameter. Deep learning techniques are utilized to optimize the networks. Compared with conventional BP, the results show that SNND achieves competitive BER performance. Moreover, the complexity and latency are much lower according to the analysis. Our future work will focus on further improvements of SNND using other decoding methods, such as [14] or [13].

Acknowledgement

This work is supported in part by NSFC under grants 61871115 and 61501116, Jiangsu Provincial NSF for Excellent Young Scholars under grant BK20180059, Huawei HIRP Flagship under grant YB201504, the Fundamental Research Funds for the Central Universities, the SRTP of Southeast University, State Key Laboratory of ASIC & System under grant 2016KF007, ICRI for MNC, and the Project Sponsored by the SRF for the Returned Overseas Chinese Scholars of MoE.

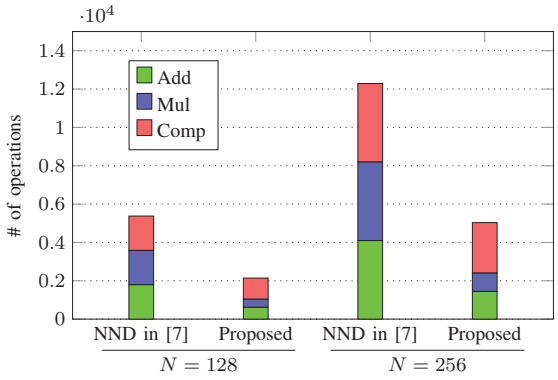


Fig. 9. Complexity comparison of one iteration for proposed SNND with single weight and polar NND in [7].

References

[1] S. Krastanov and L. Jiang, "Deep neural network probabilistic decoder for stabilizer codes," *Scientific Reports*, vol. 7, no. 1, p. 11003, 2017.

[2] H. Ye and G. Y. Li, "Initial results on deep learning for joint channel equalization and decoding," in *IEEE Vehicular Technology Conference (VTC-Fall)*, 2017, pp. 1–5.

[3] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *arXiv preprint arXiv:1802.02046*, 2018.

[4] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[5] T. Gruber, S. Cammerer, J. Hoydis, and S. t. ten Brink, "On deep learning-based channel decoding," in *Annual Conference on Information Sciences and Systems (CISS)*, 2017, pp. 1–6.

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

[7] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *IEEE International Workshop on Signal Processing Systems (SiPS)*, 2017, pp. 1–6.

[8] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016, pp. 341–346.

[9] S. Cammerer, M. Ebada, A. Elkelesh, and S. t. Brink, "Sparse graphs for belief propagation decoding of polar codes," *arXiv preprint arXiv:1712.08538*, 2017.

[10] E. Arikan, "A performance comparison of polar codes and reed-muller codes," *IEEE Communications Letters*, vol. 12, no. 6, 2008.

[11] N. Goela, S. B. Korada, and M. Gastpar, "On LP decoding of polar codes," in *IEEE Information Theory Workshop (ITW)*, 2010, pp. 1–5.

[12] E. Arikan, "Systematic polar coding," *IEEE Communications Letters*, vol. 15, no. 8, pp. 860–862, 2011.

[13] E. Nachmani, Y. Bachar, E. Marciano, D. Burshtein, and Y. Beery, "Near maximum likelihood decoding with deep learning," in *The International Zurich Seminar on Information and Communication*, 2018, pp. 40–44.

[14] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Beery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

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

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

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