Neural-Network-Optimized Degree-Specific Weights for LDPC MinSum Decoding

Linfang Wang*, Sean Chen*, Jonathan Nguyen*, Divsalar Dariush†, Richard Wesel*
*Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, California 90095
†Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109
Email: {lfwang, nguyen.j,wesel}@ucla.edu, mistystory@g.ucla.edu, Dariush.Divsalar@jpl.nasa.gov

2021 11th International Symposium on Topics in Coding (ISTC) | 978-1-6654-0943-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISTC49272.2021.9594227

***Abstract*—Neural Normalized MinSum (N-NMS) decoding delivers better frame error rate (FER) performance on linear block codes than conventional Normalized MinSum (NMS) by assigning dynamic multiplicative weights to each check-to-variable node message in each iteration. Previous N-NMS efforts primarily investigated short block codes ($N < 1000$), because the number of N-NMS parameters required to be trained scales proportionately to the number of edges in the parity check matrix and the number of iterations. This imposes an impractical memory requirement for conventional tools such as Pytorch and Tensorflow to create the neural network and store gradients. This paper provides efficient methods of training the parameters of N-NMS decoders that support longer block lengths. Specifically, this paper introduces a family of Neural 2-dimensional Normalized (N-2D-NMS) decoders with various reduced parameter sets and shows how performance varies with the parameter set selected. The N-2D-NMS decoders share weights with respect to check node and/or variable node degree. Simulation results justify a reduced parameter set, showing that the trained weights of N-NMS have a smaller value for the neurons corresponding to larger check/variable node degree. Further simulation results on a (3096,1032) Protograph-Based Raptor-Like (PBRL) code show that the N-2D-NMS decoder can achieve the same FER as N-NMS while also providing at least a 99.7% parameter reduction. Furthermore, the N-2D-NMS decoder for the (16200,7200) DVBS-2 standard LDPC code shows a lower error floor than belief propagation. Finally, this paper proposes a hybrid decoder training structure that utilizes a neural network which combines a feedforward module with a recurrent module. The decoding performance and parameter reduction of the hybrid training depends on the length of recurrent module of the neural network.**

I. Introduction

Message passing decoders are often used to decode linear block codes. Typical message passing decoders utilize belief propagation (BP), MinSum, and its variations. However, message passing decoders are sub-optimal because of the existence of cycles in the corresponding Tanner graph.

Recently, numerous works have focused on enhancing the performance of message passing decoders with the help of neural networks [1]–[14]. The neural network is created by unfolding the message passing operations of each decoding iteration [1]. Nachmani *et al.* in [1] proposed improving BP decoding by assigning unique multiplicative weights to check-to-variable messages and the channel log-likelihood (LLR) of variables in each iteration. This so-called "Neural BP (NBP)" showed better performance than BP. Continuing off this, the authors later proposed a recurrent neural network BP (RNN-BP) [3] decoder which set the edge-specific weight to be equal in each iteration. Nachmani *et al.* and Lugosch *et al.* in [1], [2], [4] proposed a Neural Normalized MinSum (N-NMS) decoder and Neural Offset MinSum (N-OMS) decoder to improve the performance of the NMS and OMS decoder.

However, with longer code lengths, these edge-specific neural decoders become impractical because the number of edges scales rapidly. A possible solution is to share one parameter across all edges that have some similar property such as across the same iteration, or connecting to the same check/variable node. For an example, Wang *et al.* proposed to assign the same parameters for each check-to-variable layer and variable-to-check layer [13], respectively. M. Lian *et. al.* in [14] considered assigning same weight to all messages in one iteration.

With these previous papers, the focus on short block length codes ($N < 1000$) may have resulted from the fact that popular deep learning research platforms, such as Pytorch and Tensorflow, require impractical amounts of memory to calculate the gradient when the block length is long. However, as demonstrated in [11], it is possible to train parameters for longer block lengths if resources are handled more efficiently. Abotabl *et al.* provided an efficient computation framework for optimizing the offset values in the N-OMS algorithm [11], and trained an OMS neural network with edge-specific weights, iteration-specific weights, and a single weight.

A primary contribution of this paper is a family of neural 2-dimensional normalized MinSum (N-2D-NMS) decoders whose weights are optimized by a neural network based on node degree. This simplification over previous approaches that optimize the weights based on node degree leads to a much simpler optimization that provides excellent FER performance while still accommodating large block lengths of practical importance. The main contributions in this paper are:

- An efficient implementation of the N-NMS architecture. This part is related to the framework in [11]. We showed that the memory issued for training long LDPC code faced by Tensorflow [15] can be mitigated by efficiently storing the check-to-variable node and variable-to-check

This research is supported by Physical Optics Corporation (POC), SA Photonics, and National Science Foundation (NSF) grant CCF-1911166. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect views of POC, SA or NSF. Research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA.

node messages. Separately, back propagation memory is also reduced by storing the gradient with respect to check-to-variable node and variable-to-check node messages only for the previous iteration rather than all iterations.
- Empirical N-NMS results that show the dynamic weights exhibit a strong correlation with check node degree, variable node degree, and iteration.
- A family of N-2D-NMS decoders with various reduced parameter sets showing how performance varies with the parameter set selected. The N-2D-NMS decoding structure is a generalization of [16] to allow variation with iteration. Simulation results on a (3096,1032) PBRL code show that N-2D-NMS decoder can achieve the same FER as N-NMS with significantly fewer parameters. A N-2D-NMS decoder trained on the (16200,7200) DVBS-2 LDPC code achieves a lower error floor than belief propagation.
- A hybrid decoding structure that combines a feedforward and recurrent structure that shows similar decoding performance as a full feedforward structure, but requires significantly fewer parameters.

The remainder of the paper is organized as follows: Sec. II derives the gradients of the loss function with respect to trainable parameters and neurons of a N-NMS neural network and proposes an efficient learning representation. Furthermore, statistics of learned weights are studied in this section. Sec. III introduces the family of N-2D-NMS decoders. Sec. IV presents and discusses our simulation results and explores a hybrid decoding structure. Sec. V concludes our work.

II. Efficient implementation of N-NMS

A. Forward Propagation

Let H be the parity check matrix of an (n, k) LDPC code, where n and k represent the codeword length and dataword length, respectively. We use v_i and c_j to denote the i^{th} variable node and j^{th} check node, respectively. In each iteration, an NMS decoder uses the same constant value to scale all check-to-variable node messages, whereas an N-NMS decoder assigns distinct multiplicative parameters for each check-to-variable message in each iteration. In the t^{th} *decoding iteration*, N-NMS updates the check-to-variable node message, $u^{(t)}_{c_j\to v_i}$, the variable-to-check node message, $l^{(t)}_{v_i\to c_j}$, and posterior of each variable node, $l^{(t)}_{v_i}$, by:

$$u^{(t)}_{c_i\to v_j} = \beta^{(t)}_{(c_i,v_j)} \times \prod_{v_{j'}\in N(c_i)/\{v_j\}} \operatorname{sgn}(l^{(t-1)}_{v_{j'}\to c_i}) \times \min_{v_{j'}\in N(c_i)/\{v_j\}} \left|(l^{(t-1)}_{v_{j'}\to c_i})\right| , \quad (1)$$

$$l^{(t)}_{v_j\to c_i} = l^{ch}_{v_i} + \sum_{c_{i'}\in N(v_j)/\{c_i\}} u^{(t)}_{c_{i'}\to v_j}, \quad (2)$$

$$l^{(t)}_{v_j} = l^{ch}_{v_i} + \sum_{c_{i'}\in N(v_j)} u^{(t)}_{c_{i'}\to v_j}. \quad (3)$$

$N(c_i)$ represents the set of the variable nodes that are connected to c_i and $N(v_j)$ represents the set of the check nodes that are connected to v_j. $l^{ch}_{v_j}$ is the LLR of channel observation of v_j. $\beta^{(t)}_{(c_i,v_j)}$ are multiplicative weights to be trained. The decoding process stops when all parity checks are satisfied or maximum iteration I_T is reached.

B. Backward Propagation

In this subsection, we derive the gradient of J with respect to the trainable weights, $\frac{\partial J}{\partial \beta^{(t)}_{(v_i,c_j)}}$, the check-to-variable message, $\frac{\partial J}{\partial u^{(t)}_{c_i\to v_j}}$, and variable-to-check message, $\frac{\partial J}{\partial l^{(t)}_{v_j\to u_i}}$. We show that in order to calculate the desired gradients, it is sufficient only to store, $l^{(t)}_{v_i}$,sgn($l^{(t)}_{v_j\to c_i}$), sgn($u^{(t)}_{c_i\to v_j}$), $\texttt{min1}^t_{c_i}$, $\texttt{min2}^t_{c_i}$, $\texttt{pos1}^t_{c_i}$ and $\texttt{pos1}^t_{c_i}$ when performing forward propagation, where

$$\texttt{min1}^t_{c_i} = \min_{v_{j'}\in N(c_i)} |l^{(t)}_{v_{j'}\to c_i}|, \quad (4)$$

$$\texttt{pos1}^t_{c_i} = \operatorname*{argmin}_{v_{j'}\in N(c_i)} |l^{(t)}_{v_{j'}\to c_i}|, \quad (5)$$

$$\texttt{min2}^t_{c_i} = \min_{v_{j'}\in N(c_i)/\{\texttt{pos1}^t_{c_i}\}} |l^{(t)}_{v_{j'}\to c_i}|, \quad (6)$$

$$\texttt{pos2}^t_{c_i} = \operatorname*{argmin}_{v_{j'}\in N(c_i)/\{\texttt{pos1}^t_{c_i}\}} |l^{(t)}_{v_{j'}\to c_i}|. \quad (7)$$

In this paper, multi-loss cross entropy [1] is used as loss function. In iteration t, $\frac{\partial J}{\partial l^{(t)}_{v_j\to u_i}}$ is updated as follows:

$$\frac{\partial J}{\partial u^{(t)}_{v_j\to c_i}} = \frac{\partial J}{\partial l^{(t)}_{v_j}} + \sum_{c_{i'}\in N(v_j)/\{c_i\}} \frac{\partial J}{\partial l^{(t)}_{c_{i'}\to v_j}}. \quad (8)$$

$\frac{\partial L}{\partial \beta^{(t)}_{c_i\to v_j}}$ is calculated by:

$$\frac{\partial J}{\partial \beta^{(t)}_{c_i\to v_j}} = u^{(t)*}_{c_i\to v_j} \frac{\partial J}{\partial u^{(t)}_{c_i\to v_j}}, \quad (9)$$

where

$$u^{(t)*}_{c_i\to v_j} = \operatorname{sgn}(u^{(t)*}_{c_i\to v_j}) \times |u^{(t)*}_{c_i\to v_j}|, \quad (10)$$

$$\operatorname{sgn}(u^{(t)*}_{c_i\to v_j}) = \prod_{v_{j'}\in N(c_i)/\{v_j\}} \operatorname{sgn}(l^{(t-1)}_{v_{j'}\to c_i}), \quad (11)$$

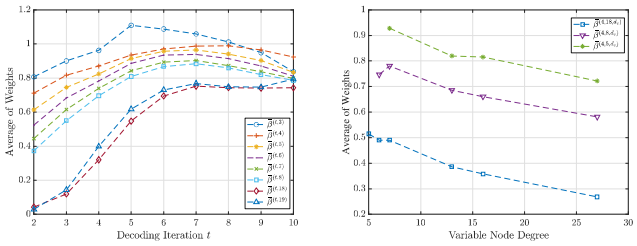
$$|u^{(t)*}_{c_i\to v_j}| = \begin{cases} \texttt{min2}^t_{c_i}, & \text{if } v_j = \texttt{pos1}^t_{c_i} \\ \texttt{min1}^t_{c_i}, & \text{otherwise} \end{cases}. \quad (12)$$

With chain rule, we obtain the following for $\frac{\partial J}{\partial |u^{(t)*}_{c_i\to v_j}|}$:

$$\frac{\partial J}{\partial |u^{(t)*}_{c_i\to v_j}|} = \operatorname{sgn}(u^{(t)*}_{c_i\to v_j}) \frac{\partial J}{\partial u^{(t)*}_{c_i\to v_j}}, \quad (13)$$

$$= \operatorname{sgn}(u^{(t)*}_{c_i\to v_j}) \beta^{(t)}_{(c_i,v_j)} \frac{\partial J}{\partial u^{(t)}_{c_i\to v_j}}. \quad (14)$$

For all variable nodes connected to check node c_i, only $\texttt{pos1}^{(t)}_{c_i}$ and $\texttt{pos2}^{(t)}_{c_i}$ receive backward information, therefore, $\frac{\partial J}{\partial l^{(t-1)}_{v_j\to c_i}}$ can be computed as follows:



(a) (b)

Fig. 1. Mean values of messages of FNNMS for a (3096,1032) PBRL code in each iteration show strong correlations to check and variable node degree.

$$\begin{cases} \operatorname{sgn}(l_{v_j\to c_i}^{(t-1)})\sum_{v_{j'}\in N(c_i)/v_j}\frac{\partial J}{\partial|u_{c_i\to v_{j'}}^{(t)*}|} & ,v_j=\texttt{pos1}_{c_i}^{(t)} \\ \operatorname{sgn}(l_{v_j\to c_i}^{(t-1)})\frac{\partial J}{\partial\left|u_{c_i\to \texttt{pos1}_{c_i}^{(t)}}^{(t)*}\right|} & ,v_j=\texttt{pos2}_{c_i}^{(t)} \\ 0 & ,\text{otherwise.} \end{cases} \tag{15}$$

Eqs. (8-15) indicate that $\frac{\partial J}{\partial u_{c_i\to v_j}^{(t)}}$ and $\frac{\partial J}{\partial l_{v_j\to c_i}^{(t)}}$ can be calculated iteratively. Therefore, back propagation does not need to store the gradients with respect to $u_{c_i\to v_j}$ and $l_{v_j\to c_i}$ for all iterations. Both Pytorch and Tensorflow store all iterations of $u_{c_i\to v_j}^{(t)}$, $l_{v_j\to c_i}^{(t)}$ and $l_{v_j}^{(t)}$, making them inefficient for this purpose. However, we showed that the neuron values in each hidden layer can be stored compactly using the parameters $l_{v_i}^{(t)}$, $\operatorname{sgn}\left(l_{v_j\to c_i}^{(t)}\right)$, $\operatorname{sgn}\left(u_{c_i\to v_j}^{(t)}\right)$, $\texttt{min1}_{c_i}^{t}$, $\texttt{min2}_{c_i}^{t}$, $\texttt{pos1}_{c_i}^{t}$ and $\texttt{pos1}_{c_i}^{t}$, which results in a significant reduction in storage requirements. Using these two strategies, we resolve the Tensorflow storage obstacle identified by [15].

C. Simulation Results

In this subsection, we use the efficient implementation described above to train the weights of N-NMS for a (3096,1032) protograph-based raptor-like (PBRL) LDPC code. The code we use is taken from [17] (in [18]). Encoded bits x are modulated by binary phase shift keying (BPSK) and transmitted through Additive White Gaussian Noise (AWGN) Channel. The N-NMS decoder is updated on a flooding schedule and the maximum number decoding iterations is 10. Define $\beta^{(t,d_c)}=\{\beta_{(c_i,v_j)}^{(t)}|\deg(c_i)=d_c\}$ and $\bar{\beta}^{(t,d_c)}$ as the mean value of $\beta^{(t,d_c)}$. Fig.1a shows $\bar{\beta}^{(t,d_c)}$ versus decoding iteration t with all possible check node degrees. Note that the iteration number starts at 2 because most of edges have 0 valued messages in the first iteration as result of puncturing. The simulation shows a clear relationship between check node degree and $\bar{\beta}$, i.e. larger check node degrees correspond to smaller $\bar{\beta}$. This difference is significant in the first few iterations. Additionally, $\bar{\beta}^{(t,d_c)}$ changes significantly in first few iterations for all check node degrees d_c.

In order to investigate the relationship between weights and variable node degree given a check node degree and iteration number, we further define $\beta^{(t,d_c,d_v)}=\{\beta_{(c_i,v_j)}^{(t)}|\deg(c_i)=d_c,\deg(v_i)=d_v\}$. We denote $\bar{\beta}^{(t,d_c,d_v)}$ to be the average value of $\beta^{(t,d_c,d_v)}$. Fig.1b gives the average value of weights corresponding to various check and variable node degrees at iteration 4. Simulation results show that, given a specific iteration t' and check node degree d_c', larger d_v' correspond to smaller $\bar{\beta}^{(t',d_c',d_v')}$.

In conclusion, the weights of N-NMS are correlated with check node degree, variable node degree, and iteration. Thus, node degrees should affect the weighting of messages on their incident edges when decoding irregular LDPC codes. Inspired by recent neural network decoders, we propose a family of N-2D-NMS decoders in this paper.

TABLE I
VARIOUS N-2D-NMS DECODERS AND REQUIRED NUMBER OF PARAMETERS PER ITERATION

Type	$\beta_*^{(t)}$	$\alpha_*^{(t)}$	The number of Required Parameters per Iteration	
			(16200,7200) DVBS-2 code	(3096,1032) PBRL code
No Weight Sharing				
0 [1]	$\beta_{(c_i,v_j)}^{(t)}$	1	$4.8*10^5$	$1.60*10^4$
Weight Sharing Based on Node Degree				
1	$\beta_{(d.(c_i),d.(v_j))}^{(t)}$	1	13	41
2	$\beta_{(deg(c_i))}^{(t)}$	$\alpha_{(deg(v_j))}^{(t)}$	8	15
3	$\beta_{(deg(c_i))}^{(t)}$	1	4	8
4	1	$\alpha_{(deg(v_j))}^{(t)}$	4	7
Weight Sharing Based on Protomatrix				
5 [19]	$\beta_{\left(\left\lfloor\frac{i}{f}\right\rfloor,\left\lfloor\frac{j}{f}\right\rfloor\right)}^{(t)}$	1	–	101
6	$\beta_{\left(\left\lfloor\frac{i}{f}\right\rfloor\right)}^{(t)}$	1	–	17
7	1	$\alpha_{\left(\left\lfloor\frac{j}{f}\right\rfloor\right)}^{(t)}$	–	25
Weight sharing based on Iteration Lian2019-jh,Abotabl2019-wt				
8	$\beta^{(t)}$	1	1	1

III. NEURAL 2D NORMALIZED MINSUM DECODERS

Based on the previous discussion, it is intuitive to consider assigning the same weights to messages with same check node degree and/or variable node degree. In this section, we propose neural 2-dimensional normalized MimSum (N-2D-NMS) decoders which has the following form:

$$\begin{aligned} u_{c_i\to v_j}^{(t)} = \beta_*^{(t)} \times \prod_{v_{j'}\in N(c_i)/\{v_j\}} \operatorname{sgn}(l_{v_{j'}\to c_i}^{(t-1)}) \\ \times \min_{v_{j'}\in N(c_i)/\{v_j\}} |(l_{v_{j'}\to c_i}^{(t-1)})| \end{aligned}, \tag{16}$$

$$l_{v_j\to c_i}^{(t)} = l_{v_i}^{ch} + \alpha_*^{(t)} \sum_{c_{i'}\in N(v_j)/\{c_i\}} u_{c_{i'}\to v_j}^{(t)}. \tag{17}$$

$\beta_*^{(t)}$ and $\alpha_*^{(t)}$ are the multiplicative weights assigned to check and variable node messages, respectively. Table I lists different types of N-2D-NMS decoders, each identified in the first

column by a type number. As a special case, we denote N-NMS as type 0. Columns 2 and 3 describe how each type assigns $\beta_*^{(t)}$ and $\alpha_*^{(t)}$, respectively. The subscript * is replaced in Table I with the information needed to identify the specific weight depending on the weight sharing methodology.

Types 1-4 assign the same weights based on node degree. In particular, Type 1 assigns the same weight to the edges that have same check node *and* variable node degree. Type 2 considers the check node degree and variable node degree separately. As a simplification, type 3 and type 4 only consider variable node degree and check node degree, respectively.

Dai. *et. al* in [19] studied weight sharing based on the edge type of MET-LDPC codes, or protograph-based codes. We also consider this metric for types 5, 6, and 7. Type 5 assigns the same weight to edges with the same edge type, i.e. edges that belong to the same position in protomatrix. In Table. I, f is the lifting factor. Types 6 and 7 assign parameters based only on the horizontal (protomatrix row) and vertical layers (protomatrix column), respectively. Finally, type 8 assigns a single weight parameter for each iteration, as in [11], [14].

To further reduce the number of parameters, we consider a hybrid training structure that utilizes a neural network combining a feedforward module with a recurrent module . The corresponding decoder uses distinct trained parameters for each of the first I' decoder iterations and reuses the same parameters for for the remaining $I_T - I'$ iterations. The motivation for the hybrid decoder is that the values of the trainable parameters change negligibly during the last few iterations, as illustrated in Sec. IV. Therefore, using the same parameters for the last few iterations doesn't cause a large performance degradation.

A (3096,1032) PBRL code and the (16200,7200) DVBS-2 [20] standard LDPC code are considered in this paper, and the number of parameters per iteration required for various N-2D-NMS decoders of these two codes are listed in column 4 and 5 in Table. I, respectively. It is shown that the number of parameters required by node-degree based weight sharing is less than that required by protomatrix based weight sharing.

IV. Simulation Results

In this section, we investigate the decoding performance of various N-2D-NMS decoders for LDPC codes with different block lengths. All encoded bits are modulated by BPSK and transmitted through the AWGN channel. The LDPC codes and optimized weights in this paper can downloaded in [17].

A. (16200,7200) DVBS-2 LDPC code

Fig. 2 gives the FER performances of various LDPC decoder for (16200,7200) DVBS-2 LDPC code. All of the decoders are *flooding* shceduled and maximum decoding iteration is 50. It is shown that N-NMS decoder outperforms BP at $1.3dB$, with a lower error floor. The N-2D-NMS decoders of types 1 and 2 have a slightly better performance than N-NMS. Type 4 outperforms type 3, because the variable node weights of investigated code have a larger dynamic range than check node weights, as shown in Fig. 2.

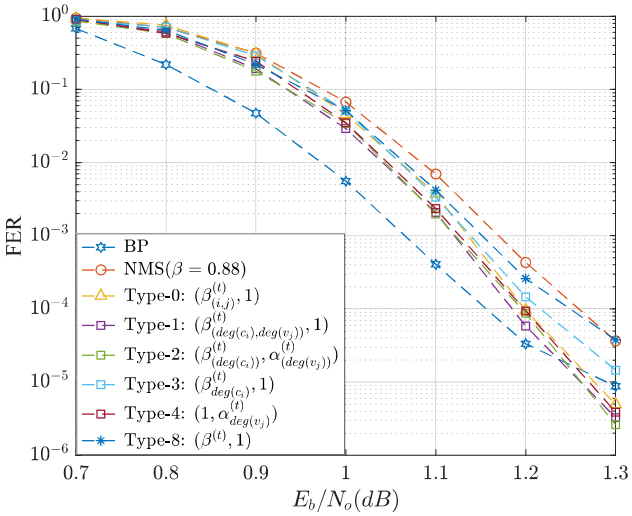


Fig. 2. FER performances of various LDPC decoder for (16200,7200) DVBS-2 LDPC code.

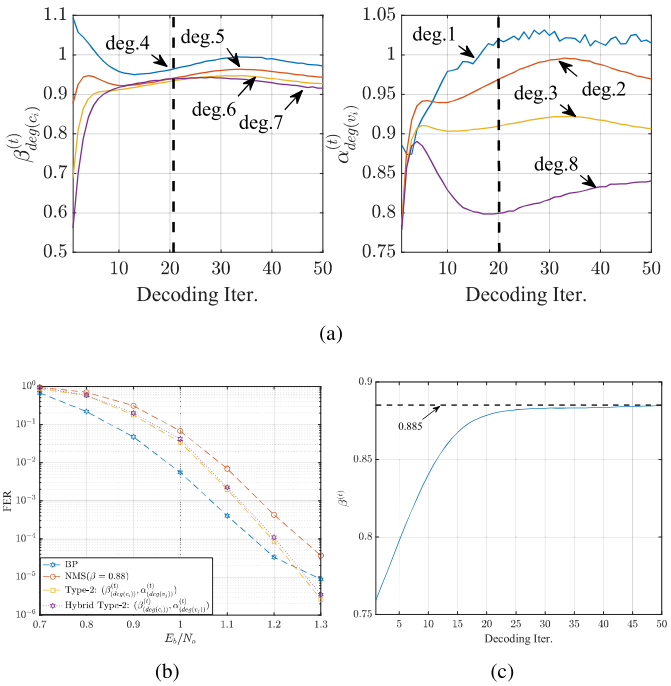


Fig. 3. Fig. (a) shows that the weights of the Type-2 N-2D-NMS decoder for the (16200,7200) DVBS-2 LDPC code only change significantly in the first 20 iterations. Fig. (b) illustrates that the Hybrid Type-2 N-2D-NMS decoder with $I' = 20$ shows comparable decoding performance to the full feedforward decoder. Fig.(c) shows that the weights of the Type-2 N-2D-NMS decoder for the (16200,7200) DVBS-2 LDPC code converge to 0.885.

Fig. 3a shows the $\beta^{(t)}_{(\deg(c_i))}$ and $\alpha^{(t)}_{(\deg(v_j))}$ of type-2 N-2D-NMS decoder, which agree with our observation in the previous section, i.e., in each decoding iteration, larger degree node corresponds to a smaller value. Besides, the weights change negligibly after iteration 20. Thus, the hybrid N-2D-NMS decoder of type 2 with $I' = 20$ delivers comparable performance to the full feedforward decoding structure, as

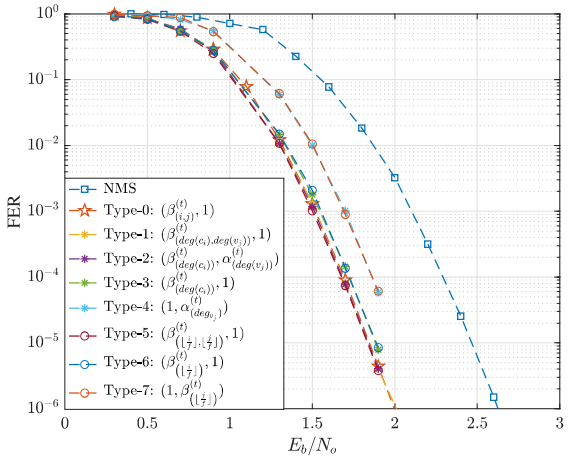


Fig. 4. FER performance for various N-2D-NMS decoders for a (3096,1032) PBRL LDPC code compared with N-NMS (type 0) and NMS.

shown in Fig. 3b. Fig. 3c shows that the parameters of type 8 converge to 0.885, which is close to the single weight of NMS decoder. As shown in Fig.2, by only assigning iteration-specific parameters, N-2D-NMS decoder of type 8 appears an early error floor at 1.20 dB.

B. (3096,1032) PBRL LDPC Code

Fig. 4 compares the FER performance of various N-2D-NMS decoders with the N-NMS (type 0) and NMS. All of the decoders are implemented using a layered schedule with a maximum of 10 decoding iterations. The simulation results show that N-NMS has more than 0.5 dB improvement over the NMS decoder. N-2D-NMS decoders of types 1-7 are also simulated. Note that types 1, 2 and 5 have the same decoding performance as the N-NMS decoder, but the number of parameters is reduced by 99.7%, 99.9% and 99.3%, respectively. Thus, weight-sharing metrics based on check and variable node degree, or based on horizontal and vertical layer deliver lossless performance with respect to N-NMS. N-2D-NMS decoders of types 4 and 6 have a degradation of around 0.05 dB compared to N-NMS. N-2D-NMS decoders of types 5 and 7 have a degradation of around 0.2 dB compared with N-NMS. Thus, for the (3096,1032) PBRL code of Fig. 4, assigning weights based only on check nodes can gain more benefit than assigning weights only based on variable node.

V. Conclusion

This paper investigates MinSum LDPC decoders for which the normalization weights are optimized by a neural network. An initial neural network assigns a different weight to every edge. The statistics of the trained parameters show that the trained parameters depend on node degree. In particular, the trained weights have a smaller value for the neurons corresponding to a larger check/variable node degree. Neural 2D normalized MinSum (N-2D-NMS) decoders are introduced in this paper with various weight-sharing techniques to reduce the number of parameters that must be trained. Simulation results on the (16200,7200) DVBS-2 standard LDPC code and a (3096,1032) PBRL code show that the N-2D-NMS decoder achieves comparable decoding performance to a N-NMS decoder but with dramatically fewer trained parameters. Furthermore, N-2D-NMS decoders can achieve a lower error floor than BP for some LDPC codes. Finally, this paper proposes a hybrid neural network with both feedforward and recurrent modules for further parameter reduction. The decoding performance and parameter reduction of the hybrid structure depends on the length of feeedforward neural network.

References

[1] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing*, Sep. 2016, pp. 341–346.

[2] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *2017 IEEE Inter. Symp. on Info. Theory (ISIT)*, Jun. 2017, pp. 1361–1365.

[3] E. Nachmani, E. Marciano, D. Burshtein, and Y. Be'ery, "RNN decoding of linear block codes," *CoRR*, vol. abs/1702.07560, 2017. [Online]. Available: http://arxiv.org/abs/1702.07560

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

[5] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 1, pp. 144–159, Feb. 2018.

[6] X. Wu, M. Jiang, and C. Zhao, "Decoding optimization for 5G LDPC codes by machine learning," *IEEE Access*, vol. 6, pp. 50 179–50 186, 2018.

[7] L. Lugosch and W. J. Gross, "Learning from the syndrome," in *2018 52nd Asilomar Conf. on Signals, Systems, and Computers*, Oct. 2018, pp. 594–598.

[8] W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang, "Performance evaluation of channel decoding with deep neural networks," in *2018 IEEE Inter. Conf. on Comm. (ICC)*, May 2018, pp. 1–6.

[9] X. Xiao, B. Vasic, R. Tandon, and S. Lin, "Finite alphabet iterative decoding of LDPC codes with coarsely quantized neural networks," in *2019 IEEE Global Comm. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[10] C. Deng and S. L. Bo Yuan, "Reduced-complexity deep neural network-aided channel code decoder: A case study for BCH decoder," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 1468–1472.

[11] A. Abotabl, J. H. Bae, and K. Song, "Offset min-sum optimization for general decoding scheduling: A deep learning approach," in *2019 IEEE 90th Vehicular Tech. Conf. (VTC2019-Fall)*, Sep. 2019, pp. 1–5.

[12] A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. G. i Amat, "Pruning and quantizing neural belief propagation decoders," *IEEE Journal on Selected Areas in Communications*, 2020.

[13] Q. Wang, S. Wang, H. Fang, L. Chen, L. Chen, and Y. Guo, "A Model-Driven deep learning method for normalized Min-Sum LDPC decoding," in *2020 IEEE Inter. Conf. on Com. Workshops*, Jun. 2020, pp. 1–6.

[14] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned Belief-Propagation decoding with simple scaling and SNR adaptation," in *2019 IEEE Inter. Symp. on Information Theory (ISIT)*, Jul. 2019, pp. 161–165.

[15] L. P. Lugosch, "Learning algorithms for error correctio," 2018, master thesis, McGill University.

[16] Juntan Zhang, M. Fossorier, Daqing Gu, and Jinyun Zhang, "Improved min-sum decoding of ldpc codes using 2-dimensional normalization," in *IEEE Global Comm. Conf., 2005.*, vol. 3, 2005, pp. 1187–1192.

[17] "UCLA communications systems laboratory," http://www.seas.ucla.edu/csl/#/publications/published-codes-and-design-tools.

[18] T. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like ldpc codes," *IEEE Trans. on Comm.*, vol. 63, no. 5, pp. 1522–1532, 2015.

[19] J. Dai, K. Tan, Z. Si, K. Niu, M. Chen, H. V. Poor, and S. Cui, "Learning to decode protograph ldpc codes," *arXiv:2102.03828*, 2021.

[20] ETSI EN 302 307, "Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2)."