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

***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 message in each iteration. Previous N-NMS efforts have primarily investigated short-length block codes ($N < 1000$), because the number of N-NMS parameters to be trained is proportional to the number of edges in the parity check matrix and the number of iterations, which imposes am impractical memory requirement when Pytorch or Tensorflow is used for training. This paper provides efficient approaches to training parameters of N-NMS that support N-NMS for longer block lengths. Specifically, this paper introduces a family of neural 2-dimensional normalized (N-2D-NMS) decoders with 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 this approach, showing that the trained weights of N-NMS have a strong correlation to the check node degree, variable node degree, and iteration number. Further simulation results on the (3096,1032) protograph-based raptor-like (PBRL) code show that N-2D-NMS decoder can achieve the same FER as N-NMS with significantly fewer parameters required. The N-2D-NMS decoder for a (16200,7200) DVBS-2 standard LDPC code shows a lower error floor than belief propagation. Finally, a hybrid decoding structure combining a feedforward structure with a recurrent structure is proposed in this paper. The hybrid structure shows similar decoding performance to full feedforward structure, but requires significantly fewer parameters.**

I. Introduction

Message passing decoders are often used for linear block code decoding. Typical message passing decoders utilize belief propagation (BP), MinSum, and its variations such as normalized MinSum (NMS) and offset MinSum (OMS). However, message passing decoders are sub-optimal because of the existence of cycles in the parity check matrix.

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. The so-called "Neural BP (NBP)" has shown better performance than BP. The authors further 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.

As the code length gets longer, these edge-specific neural decoders become impractical because the number of edges scales quickly. One solution is to share one parameter with edges that have same properties, such as in the same iteration, or connecting same check/variable node. For an example, Wang *et al.* proposed to assign 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.

Besides, most previous work has focused on codes with short block lengths ($N < 1000$). The focus on short codes may result from the fact that popular deep learning research platforms, such as Pytorch and Tensorflow, require large amounts of memory to calculate the gradient when the block length is long. However, as pointed in [11], it is possible to train the 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) decoder whose weights are optimized by a neural network based on node degree. This simplification over previous approaches that separately optimize the weight for each edge 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 based on the gradients in backpropagation. This part is related to the framework in [11]. We show that backpropagation can be conducted in an iterative way, and

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.

the parameters used to implement backpropagation can be stored efficiently. This approach solves the memory issue faced by Tensorflow [15] and facilitates an efficient C++ implementation simple enough to train without a GPU.

- Empirical N-NMS results demonstrating that dynamic weights show 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 the (3096,1032) PBRL code show that N-2D-NMS decoder can achieve the same FER as N-NMS with significantly fewer parameters. The N-2D-NMS decoder for a (16200,7200) DVBS-2 LDPC code achieves a lower error floor than belief propagation.
- A hybrid decoding structure that combines a feedforwad structure with a 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 learnable parameters and neurons of a N-NMS neural network and proposes an efficient learning representation. Statistics of learned weights are studied in this section. Sec. III introduces a 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 a (n, k) LDPC code. We use v_i and c_j to denote the i^{th} variable node and j^{th} check node, respectively. N-NMS resembles NMS except assigning 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:

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

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

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

$N(c_i)$ ($N(v_j)$) represents the set of the variable nodes (check nodes) that are connected to $c_i(v_j)$. $l^{ch}_{v_j}$ 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 learnable 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, $\frac{\partial J}{\partial l^{(t)}_{v_j\to u_i}}$. Just like forward propagation, $\frac{\partial J}{\partial u^{(t)}_{c_i\to v_j}}$ and $\frac{\partial J}{\partial l^{(t)}_{v_j\to u_i}}$ are traced back iteratively. We show that in order to get 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

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

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

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

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

In this paper, multi-loss cross entropy [1] is used as loss function. Denote loss by J and assume all-zero codewords are transmitted, it is straightforward to calculate $\frac{\partial J}{\partial l^{(t)}_{v_i}}$:

$$\frac{\partial J}{\partial l^{(t)}_{v_i}} = -\frac{1}{nI_T}\sigma(-l^{(t)}_{v_i}). \quad (8)$$

We initialize $\frac{\partial J}{\partial l^{(I_T)}_{vi\to c_j}} = 0$ for (c_i, v_j) pairs whose $H(i, j) = 1$.

In iteration t, $\frac{\partial J}{\partial l^{(t)}_{v_j\to u_i}}$ is updated based on "back propagation through time" [17]:

$$\frac{\partial J}{\partial u^{(t)}_{v_j\to c_i}} = \frac{\partial J}{\partial l^{(t)}_{v_j}} + \sum_{c_{i'}\in N(v_j)/\{c_i\}} \frac{\partial J}{\partial l^{(t)}_{c_{i'}\to v_j}}. \quad (9)$$

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

$$\frac{\partial J}{\partial \beta^{(t)}_{c_i\to v_j}} = u^{(t)*}_{c_i\to v_j}\frac{\partial J}{\partial u^{(t)}_{c_i\to v_j}}, \quad (10)$$

where

$$u^{(t)*}_{c_i\to v_j} = \text{sgn}(u^{(t)*}_{c_i\to v_j}) \times |u^{(t)*}_{c_i\to v_j}|, \quad (11)$$

$$\text{sgn}(u^{(t)*}_{c_i\to v_j}) = \prod_{v_{j'}\in N(c_i)/\{v_j\}} \text{sgn}(l^{(t-1)}_{v_{j'}\to c_i}), \quad (12)$$

$$|u^{(t)*}_{c_i\to v_j}| = \begin{cases} \texttt{min2}^t_{c_i}, & \text{if } v_j = \texttt{pos1}^t_{c_i} \\ \texttt{min1}^t_{c_i}, & \text{otherwise} \end{cases}. \quad (13)$$

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

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

$$= \text{sgn}(u^{(t)*}_{c_i\to v_j})\beta^{(t)}_{(c_i,v_j)}\frac{\partial J}{\partial u^{(t)}_{c_i\to v_j}}. \quad (15)$$

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

$$\begin{cases} \text{sgn}(l_{v_j \to c_i}^{(t-1)}) \sum_{v_{j'} \in N(c_i)/\{v_j\}} \frac{\partial J}{\partial |u_{c_i \to v_{j'}}^{(t)*}|} & \text{, if } v_j = \texttt{pos1}_{c_i}^{(t)} \\ \text{sgn}(l_{v_j \to c_i}^{(t-1)}) \frac{\partial J}{\partial \left| u_{c_i \to \texttt{pos1}_{c_i}^{(t)}}^{(t)*} \right|} & \text{, if } v_j = \texttt{pos2}_{c_i}^{(t)} \\ 0 & \text{, otherwise.} \end{cases} \tag{16}$$

Eq.(9)-(16) indicate that $\frac{\partial J}{\partial u_{c_i \to v_j}^{(t)}}$ and $\frac{\partial J}{\partial l_{v_j \to c_i}^{(t)}}$ are calculated in a message passing manner. As a result, the full N-NMS neural network is not necessarily to be built. Besides, the neuron values in each hidden layer can be stored efficiently using Eq.(4)-(7). Both solves memory issue faced by Tensorflow, as pointed in [15]. The training work in this paper is done by C++ with only CPU computation nodes. Finally, we use stochastic gradient decent method with Adam optimizer to update $\beta_{(c_i,v_j)}^{(t)}$ in each *training iteration*.

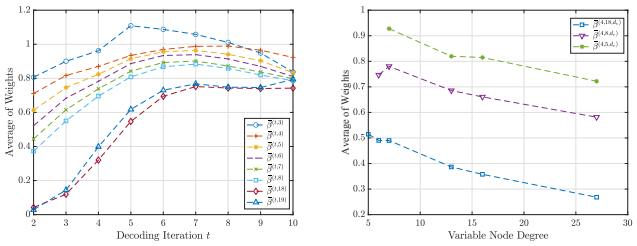
C. Simulation Results

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 [18] (in [19]). Encoded bits x are modulated by BPSK and transmitted through additive white Gaussian noise channel (AWGNC). The N-NMS decoder is flooding scheduled and maximum decoding iteration 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 gives $\bar{\beta}^{(t,d_c)}$ versus decoding iteration t with all possible check node degrees. Note that iteration number starts from 2 because most of edges have 0 messages in the first iteration, which is a result of puncturing. The simulation shows a clear relationship between check node degree and $\bar{\beta}$, i.e. the larger check node degree corresponds to a smaller $\bar{\beta}$, this difference is significant in the first few iterations. Also, for each possible d_c, $\bar{\beta}^{(t,d_c)}$ changes significantly in first few iterations. In short conclusion, Fig. 1a shows that $\beta_{(c_i,v_j)}^{(t)}$ has a strong correlation with check node degree and iteration.

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 $\bar{\beta}^{(t,d_c,d_v)}$ by the average value of $\beta^{(t,d_c,d_v)}$. Fig.1b gives the average of weights corresponding to different check node degree and variable node degree in iteration 4. Simulation result shows that given a specific iteration t' and check node degree d_c', the larger d_v' corresponds to a 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 of irregular LDPC code.



(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.

Inspired by recent neural network decoders, we propose a family of N-2D-NMS decoders in this paper.

III. Neural 2D Normalized MinSum Decoders

Based on the previous discussion, it is intuitive to consider assigning 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:

$$\begin{aligned} u_{c_i \to v_j}^{(t)} = \beta_*^{(t)} \times & \prod_{v_{j'} \in N(c_i)/\{v_j\}} \text{sgn}(l_{v_{j'} \to c_i}^{(t-1)}) \\ & \times \min_{v_{j'} \in N(c_i)/\{v_j\}} |(l_{v_{j'} \to c_i}^{(t-1)})| \end{aligned}, \tag{17}$$

$$l_{v_j \to c_i}^{(t)} = l_{v_i}^{ch} + \alpha_*^{(t)} \sum_{c_{i'} \in N(v_j)/\{c_i\}} u_{c_{i'} \to v_j}^{(t)}. \tag{18}$$

The $\beta_*^{(t)}$ and $\alpha_*^{(t)}$ are the multiplicative weights assigned to check and variable node messages, respectively. The subscripts indicate weight sharing metric. Table. I lists different types of N-2D-NMS decoders. As a special case, we denote NNMS as type-0.

Type-1 to type-4 assign the same weights based on node degree. Type-1 N-2D-NMS assigns same weights 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 [20] studied weight sharing based on edge type of MET-LDPC code, or protograph-based code. We also consider this metric in paper, which is given by type-5, 6 and 7. Type-5 assigns same weights to the edges with same edge-type, i.e. the edges belonged to the same position in protomatrix. f is lifting factor. In this paper, we further consider its simplifications, type-6 and-type 7, which assign parameters only based on horizontal layers and vertical layers, respectively. Finally, type 8 decoder assigns iteration-distinct parameters, this simple weight sharing schemes have been considered for previous literature [11], [14].

Table.I gives the number of parameters per iteration required for various N-2D-NMS decoders. The (3096,1032) PBRL code

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

type	$\beta_*^{(t)}$	$\alpha_*^{(t)}$	(16200,7200) DVBS-2 code	(3096,1032) PBRL code
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[20]	$\beta^{(t)}_{\left(\lfloor \frac{i}{f} \rfloor, \lfloor \frac{j}{f} \rfloor\right)}$	1	–	101
6	$\beta^{(t)}_{\left(\lfloor \frac{i}{f} \rfloor\right)}$	1	–	17
7	1	$\beta^{(t)}_{\left(\lfloor \frac{j}{f} \rfloor\right)}$	–	25
Weight sharing based on Iteration [11], [14]				
8	$\beta^{(t)}$	1	1	1

and (16200,7200) DVBS-2 [21] standard LDPC code are considered in this paper. 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. Based on the simulation results given in Sec. IV, the two weight sharing schemes deliver same error correction performance.

IV. SIMULATION RESULT

In this section, we investigate the decoding performance of N-2D-NMS decoders for the LDPC codes with different block length. All encoded bits are modulated by BPSK and transmitted through AWGNC. The LDPC codes and optimized weights in this paper can be found and downloaded in [18].

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 the decoders are implemented in a *flooding way* and maximum decoding iteration is 50. It is shown that N-NMS decoder outperforms BP at $1.3dB$, with a lower error floor. Type-1 and type-2 N-2D-NMS decoders have a slightly better performance than N-NMS. As two simplifications of type-2 N-2D-NMS, type 4 decoder outperforms type 3, because the variable node weights of investigated code have a larger dynamic range than check node weights, as shown in Fig.3a.

Fig. 3a shows the $\beta_{(\deg(c_i))}^{(t)}$ and $\alpha_{(\deg(v_j))}^{(t)}$ of type-2 decoder. The trained values agree with our observation in the previous section, i.e., in each decoding iteration, larger degree node corresponds to a smaller value. Note that the parameters only changes greatly in the first 20 iterations, and after that the

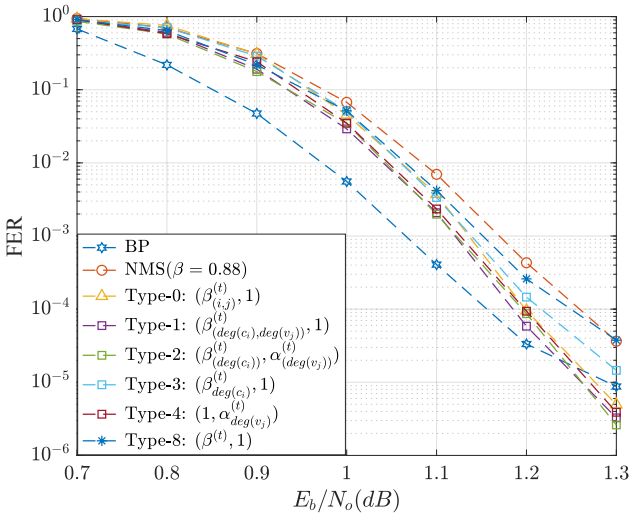
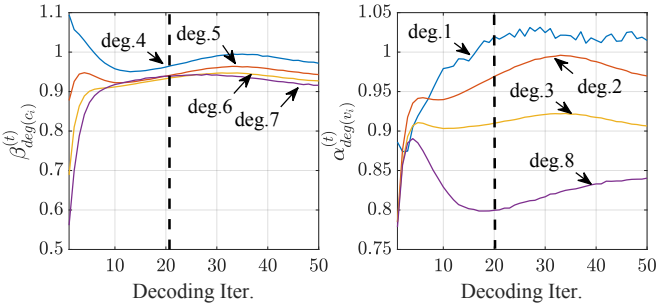
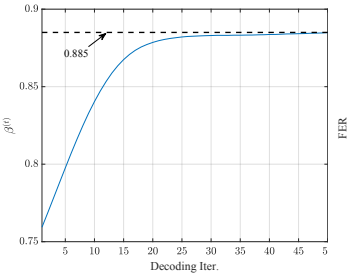


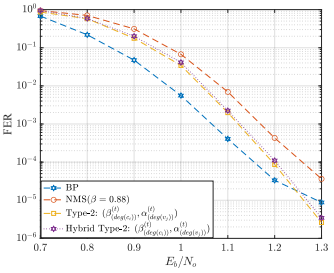
Fig. 2. FER performance N-2D-NMS decoders for a (3096,1032) PBRL LDPC code.



(a) $\beta_{deg(c_i)}^{(t)}$ and $\alpha_{deg(v_j)}^{(t)}$ of type-2 N-2DNMS decoder.



(b) $\beta^{(t)}$ of type-8 N-2D-NMS.



(c) FER performance of hybrid type-2 N-2D-NMS decoder

Fig. 3. Trained weights of N-2D-NMS decoders for a (16200,7200) DVBS-2 code only changes in significantly in the first 20 iterations. The hybrid type-2 N-2D-NMS decoder with $I' = 20$ shows a comparable decoding performance to type-2 decoder, with 60% parameters reduction.

change is minor. It directly leads to a combination of a feedforward neural network with a recurrent neural network, meaning that parameters are updated in the first I' iterations and the last $I_T - I'$ iterations reuse the parameters in iteration I'. We call this structure as *hybrid* N-2D-NMS decoder. Fig. 3c shows that hybrid type-2 N-2D-NMS wiht $I'_T = 20$ has comparable decoding performance with full feedforward decoding structure, with 60% parameter reduction.

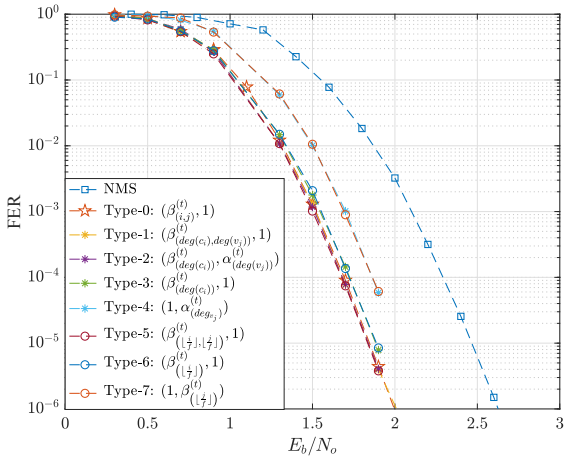


Fig. 4. FER performance N-2D-NMS decoders for a (3096,1032) PBRL LDPC code.

Fig. 3b shows that the parameters of type-8 decoder 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, type-8 decoder appears an early error floor at $1.20dB$.

B. (3096,1032) PBRL LDPC Code

Fig.4 gives the FER performance of N-2D-NMS decoders. All the decoder are implemented in a *layered way* and maximum decoding iteration is 10. The simulation results show that NNMS (i.e. type 0) has a more than $0.5dB$ improvement than NMS decoder. Type 1-7 decoders are also simulated. Note that type 1,2 and 5 have same performance with NNMS decoder, with much smaller number of parameters required. It is shown that weight sharing metrics based on check and variable node degree, or based on horizontal and vertical layer delivers lossless performance. Type-4 and 6 decoders have a degradation around $0.05dB$ compared with type 0, type-5 and 7 have a degradation around $0.2dB$ compared type 0. It can be seen that, for (3096,1032) PBRL code, assigning weights only based on check nodes can gain more benefit than assigning weights only based on variable node.

V. Conclusion

This paper presents MinSum LDPC decoders for which the normalization weights are optimized by a neural network. An initial neural network allows a different weight for every edge. The statistics of the trained parameters show a strong correlation with check node degree, variable node degree and iteration. A family of neural 2D normized MinSum (N-2D-NMS) decoders are introduced in this paper with different parameter reductions and decoding performances. Simulation results on a (16200,7200) DVBS-2 standard LDPC code and (3096,1032) PBRL code show that N-2D-NMS decoder can have same decoding performance as NNMS decoder and achieves a lower error floor than BP. Finally, a hybrid decoding structure combining feedforward structure with recurrent structure is proposed in this paper. The hybrid structure shows similar decoding performance to full feedforward structure, with less parameters required.

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 International Symposium on Information 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. Signal Process.*, 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 International Conference on Communications (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 Communications Conference (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 International Conference on Communications Workshops (ICC 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 International Symposium 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 *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, vol. 3, 2005, pp. 1187–1192.

[17] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Netw.*, vol. 1, no. 4, pp. 339–356, Jan. 1988.

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

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

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

[21] "Specifications," https://dvb.org/specifications/, Sep. 2019, accessed: 2021-5-4.