ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Performance Enhancement of Low-Density Parity-Check Decoder Using Neural Network Optimized Parameters

Amany Sabah Hassan, Mohammed Abdullah Hussein ElSheikh

Department of Electrical Engineering, College of Engineering, University of Sulaymaniyah, Sulaymaniyah, Iraq.

ABSTRACT

Low-density parity-check (LDPC) codes are of prime importance in achieving near-Shannon capacity in current communication systems. However, the optimal decoding process for LDPC codes is computationally intensive. This paper presents a neural normalized min-sum (NNMS) decoding network that improves error correction capabilities with minimal computational overhead. A weight-sharing approach is adopted in the NNMS model, in which the correction factors (α, β) are shared across nodes within a single layer. This approach decreases the total number of parameters in the model compared to traditional neural decoders, making it easier for hardware implementation. The NNMS model is tested using a (576, 432) LDPC code for an additive white Gaussian noise channel with binary phase shift keying modulation. Simulation results show that the NNMS model outperforms traditional normalized min-sum (NMS) decoders. At a signal-to-noise ratio of 5 dB, the NNMS model has a bit-error rate (BER) of 1.0×10^{-7}, whereas traditional NMS models have a less steep slope. In particular, the NNMS model has a coding gain of 1.25 dB compared to the traditional NMS model at a BER threshold of 1.0×10^{-3}. This shows that the NNMS model is an efficient solution for high-performance, real-time digital communication systems.

Index Terms: Low-Density Parity-Check Codes, Model-Driven Deep Learning, Normalized Min-Sum, Bit Error Rate, Channel Coding

1. INTRODUCTION

Gallager first presented low-density parity-check (LDPC) codes as a type of linear block code that could accomplish dependable communication with sparse parity-check matrices and low decoding error probability in his groundbreaking doctoral dissertation in the early 1960s. Gallager showed that LDPC codes can get close to the Shannon capacity with realistic decoding complexity when decoded using probabilistic iterative decoding schemes. Due to the high computational complexity of their decoding algorithms, LDPC codes' practical adoption was initially restricted despite their excellent theoretical properties [1], [2].

LDPC codes have received new attention and have been included in numerous advancements in hardware technology and digital communication systems. However, because the optimal sum-product algorithm (SPA) for LDPC decoding necessitates complex arithmetic operations, its hardware implementation is costly in terms of area and power consumption [3]. To solve this issue, simpler methods have been proposed, such as the min-sum (MS) algorithm and its modifications, which provide less complexity at the expense of some speed reduction [4].

Access this article online

DOI:10.21928/uhdjst.v10n2y2026.pp11-21

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: Amany Sabah Hassan, Department of Electrical Engineering, College of Engineering, University of Sulaymaniyah, Sulaymaniyah, Iraq. E-mail: amany.sabah.hassan@gmail.com

Received: 06-04-2026 Accepted: 31-05-2026 Published: 05-07-2026

A digital communication system's block diagram is shown in Fig. 1. The data to be transmitted are generated at the transmitter side by the source of information, and they are first processed by the source encoding stage to eliminate redundancy and effectively represent the data. After that, the encoded data are sent to the channel encoder, which adds controlled redundancy to allow error detection and correction at the recipient. The modulator then converts the encoded bits into symbols before sending them over the communication channel, where noise, interference, and fading may cause distortions [5].

To extract the soft or hard information from the noisy channel output, the demodulator at the receiver side first processes the received signal, which is called demodulation. The primary focus of this work is the channel decoder, which uses LDPC decoding algorithms to play a critical role in recovering the original transmitted data by correcting channel-induced errors. Finally, before the information is sent to the destination, source decoding reconstructs the original information sequence. This block diagram emphasizes how crucial effective channel decoding methods are to producing dependable and effective digital communication systems [6].

A key factor in lowering decoding complexity while preserving effective error-correction performance has been the graphical representation of error-correcting codes [7]. LDPC codes and product codes are generalized by Tanner's recursive graph-based framework, in which the decoding process is broken down into local computations carried out on connected subcodes through a bipartite graph [8]. LDPC codes have been adopted in many contemporary standards for wireless and igh-speed communication systems. To reduce the complication of the belief propagation (BP) decoding procedure for LDPC codes, several approximation-based algorithms were proposed. Among the approximation-based algorithms, the MS algorithm simplifies the complexity of each of the check node (CN) processing in the decoding algorithm. However, this simplification leads to a performance degradation compared to the performance obtained by the BP decoding algorithm [9]. To overcome this performance degradation and achieve performance close to the performance obtained by the BP decoding algorithm at the equivalent time, the normalized MS (NMS) algorithm was proposed. The normalization factor in the NMS algorithm scales the minimum value in the CN update [10].

Deep neural networks [11] have demonstrated considerable advancement and ability to learn difficult non-linear relationships and optimize parameters in a system where analytical modeling cannot be accomplished easily or is impractical [12]. Deep learning techniques are nearly optimal for solving complex decision-making/optimization applications using information learned from data, as evidenced by their capabilities to successfully solve complex problems such as search [13].

The recent advent of deep learning approaches has truly transformed how we can create physical layer communication systems [14]. One such way is applying model-driven deep learning to expand iterative algorithms into neural networks [15], which has produced some very positive results for improving channel decoding performance.

However, a large number of trainable weights (parameters) relative to the number of edges creates difficulties in doing hardware implementations for long block codes. Although substantial research has recently been conducted to improve this situation, there are still significant challenges to address in terms of developing an architecture that can make use of many of the benefits of neural networks while still providing low hardware overhead compared to using a fully-connected architecture [15].

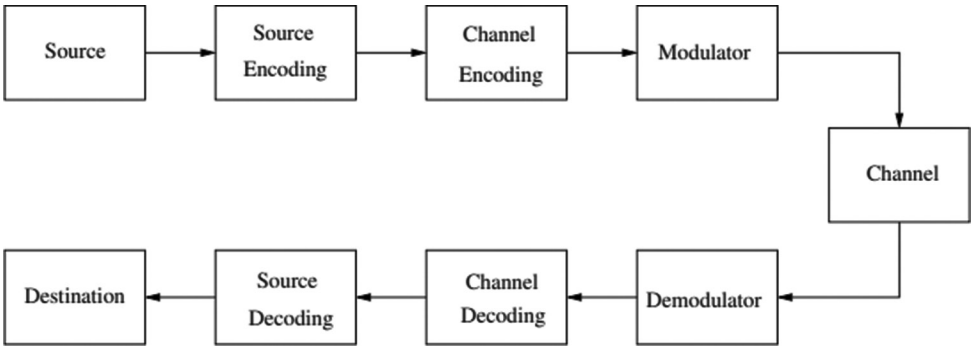

Fig. 1. Digital communication system with channel decoding [5].

To achieve high decoding accuracy with low computational complexity for long LDPC codes, we propose a Neural Normalized Min-Sum (NNMS) decoding network. To speed up the decoding process, the iterative aspect of the decoding procedure between CNs and variable nodes (VNs) is spread out into a feed-forward propagation network, and with the use of the equivalent correction factors (α, β) in every layer of the neural network to decrease the number of learnable correction factors. In addition, we demonstrate the effectiveness of the suggested NNMS approach in decoding LDPC long codes.

This article summarizes the principal contributions of this research work as follows:

1.1. Novel Network Architecture

The authors proposed a model-driven neural network (NNMS) decoder that creates a deep feed-forward network by unfolding the iterative nature of conventional NMS (CNMS) decoding algorithms, thus allowing all the parameters to be optimized using an end-to-end approach.

1.2. Weight Sharing Mechanism

Unlike traditional neural network-based decoders that assign unique weights to each edge of a Tanner graph, the authors' NNMS decoder shares its correction factors (α and β) across all nodes within a given layer of the decoder architecture. This results in a reduction of the dependency of complexity with respect to the code length, as the computational complexity becomes dependent on only the number of iterations required for convergence. This feature also makes the authors' NNMS decoder highly beneficial for hardware implementations.

1.3. Performance Evaluation

The authors conducted numerous simulations demonstrating that the NNMS decodes with better performance than traditional NMS decoding methods and provides up to 1.25 dB of coding gain measured at a bit-error rate (BER) of 1.0×10^{-3} for a (576, 432) LDPC code using a Gaussian noise channel.

1.4. Efficiency Evaluation

The authors established that their proposed method retains very low computation latency and minimal multiplicative overhead as compared to conventional neural network-based decoders by conducting a theoretical analysis of the complexity of the proposed decoder architecture.

This paper will be presented according to the following plan: Section 2 describes the networked model as well as mathematical models for the typical NMS decoder and the proposed NNMS architecture; Section 3 describes the laboratory environment, training hyperparameters, and simulation results. In addition, a comparison is provided between this method and traditional approaches; Section 4 addresses the implications from these findings and future direction based upon research done thus far; and Section 5 summarizes the paper and provides a path forward for additional investigation.

2. NETWORK MODEL AND METHODS

The LDPC codes are a type of linear block code with a sparse parity check matrix denoted as H that contains very few "1's". N bits for matrix (N,k) LDPC codes are made up of (N−k) check bits and (k) information bits. Next, the LDPC encoder encodes the k-bit information x using the formula u = xG, where G is the generation matrix that corresponds to the parity check matrix H [16]. Finally, we have the N-bit codeword u. The NMS decoding technique and model-driven deep learning networks serve as the foundation for our suggested LDPC long code decoding networks.

The following notations are used throughout this paper:

- N: Length of the codeword.
- K: Length of the information bits.
- H: Parity-check matrix of size (N-K)×N.
- V(i): Set of CNs connected to VN.
- C(j): Set of VNs connected to CN.
- y: Received symbol vector from the channel.
- σ^2: Noise variance of the additive white Gaussian noise (AWGN) channel.
- L(u_i): Initial channel log-likelihood ratio (LLR) for bit i.
- L_{rji}: Extrinsic message sent from CN j to VN i.
- L_{qij}: Extrinsic message sent from VN i to CN j.
- L(Q_i): Log-likelihood LLR value of the VN i
- Oi: estimated bit value.
- αt, βt: Learnable correction factors at iteration t.
- T_{max}: maximum number of decoding iterations.

2.1. NMS LDPC Decoder

Tanner graphs are bidirectional graphs composed of three parts: CNs, VNs, CNs and VNs, respectively, and the edges that link CNs and VNs. Squares and rings, respectively, are used to represent VNi and CNs CNj at each pair of nodes. If H(i,j) equals "1," the i-th VN is linked to the j-th CN; otherwise, it is not. In this sense, the Tanner graph [17] can represent the LDPC check matrix H. The CNMS technique is favored for the hardware implementation since it can pass

through iterative processing over the Tanner graph. Fig. 2 depicts the LDPC decoding scheme's architecture.

The following processes are used to propagate and update the CN-to-VN messages $L(r_{ji})$ and the VN-to-CN messages $L(q_{ij})$, where $L(u_i)$ is the received channel LLR corresponding to bit i.

Step 1: Use the received symbol y for initialization to compute the received channel LLR

$$L\left(q_{ij}\right)=L\left(u_i\right)=\ln\left(\frac{P\left(u_i=0\,|y_i\right)}{P\left(u_i=1\,|y_i\right)}\right)=\frac{2y_i}{\sigma^2} \quad (1)$$

Where σ^2 is the noise variance in a channel with AWGN.

Step 2: Compute the messages from CN to VN consuming

$$L\left(r_{\{ji\}}\right)=\alpha\prod_{i'\in V(j)\setminus i}sign\left(L\left(q_{\{i'j\}}\right)\right) \min_{i'\in V(j)\setminus i}\left|L\left(q_{\{i'j\}}\right)\right| \quad (2)$$

Where V_j is the collection of column locations of the "1" in the j-th row (signifies the set of VNs linked to CN_j, α is the correction factor, and V_{ji} is V_j without VN_i.

Step 3: Determine the messages from VN-to-CN using

$$L\left(q_{ij}\right)=L\left(u_i\right)+\sum_{j'\in Ci\setminus j}L\left(r_{j'i}\right) \quad (3)$$

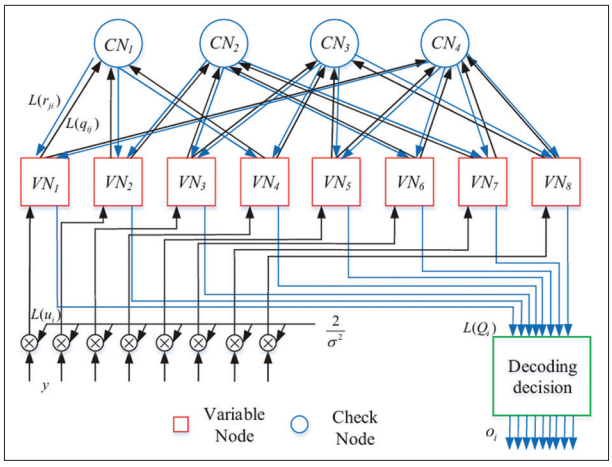

Fig. 2. The conventional normalized min-sum low-density parity-check decoding scheme.s architecture.

Where $C_{i\setminus j}$ denotes C_i excluding CN, CNj, and C_i represents the set of row locations of the "1" in the i-th column (represents the set of CNs connected to VN_i).

Step 4: Compute the log-likelihood LLR value $L(Qi)$ of the VN_i

$$L\left(Q_i\right)=L\left(u_i\right)+\sum_{j\in C(i)}L\left(r_{ji}\right) \quad (4)$$

For each decoding iteration, continue steps 2, 3, and 4 until the convergence requirements are satisfied or the maximum number of iterations is reached. Next, the VNi's estimated bit value oi is determined by

$$o_i=\begin{Bmatrix}1, L\left(Q_i\right)\le 0\\ 0, L\left(Q_i\right)>0\end{Bmatrix} \quad (5)$$

2.2. Model-based Techniques for Deep Learning

The majority of deep learning networks rely on data and do not take into account the task's mathematical model, which employs a dark box and large amounts of data to train the neural network parameters and structure. It is impossible to analyze and forecast the algorithm's structure due to a lack of knowledge about the connection between the algorithm and the network topology. Furthermore, getting a lot of label data is difficult [18].

The model, algorithm, and network components of the model-driven deep learning approach are an effective way to solve this issue. The model and algorithm of a certain task can be used to develop an intelligible network. For instance, a signal flow diagram can be created by unfolding an iterative method.

A deep neural network is then created using back propagation to train a number of learnable parameters [19]. Therefore, this method can be used to correct for the errors brought about by an inaccurate model and preset parameters. In addition, there are many other benefits to the model-driven deep learning technique, including reduced training data, a lower risk of overfitting, and quick implementation. To achieve this, our task is to learn and compensate for the uncertainty by backpropagation in order to unfold the CNMS LDPC algorithm in Fig. 3 to a neural network [20].

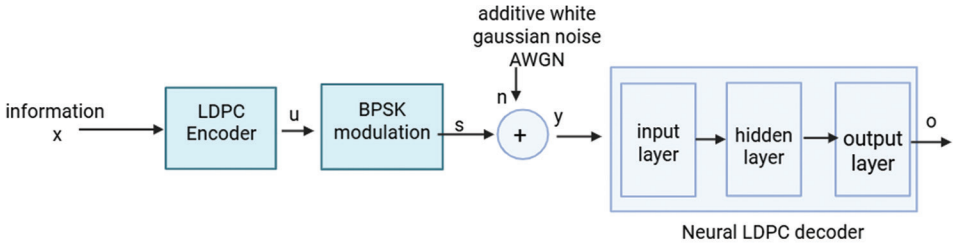

Fig. 3. Low-density parity-check decoder block diagram with the feed-forward neural network.

2.3. NNMS LDPC Decoding Network

As shown in Fig. 3, we propose the block diagram of the suggested NNMS LDPC decoding network scheme. The N-bit codeword u is created by encoding the k-bit information bits x. It is then converted into a symbol vector **s** via a BPSK modulator and sent via an AWGN channel.

The deep feed-forward neural network receives the LLR values, which are computed at the receiver using the receiver signal $y = s + n$. The receiver side detailed signal flow diagram of the proposed NNMS decoding network is displayed in Fig. 3.

The repeated decoding process is unfolded to construct a forward-propagation network. The parity check matrix H determines the edges of CNs and VNs links in the Tanner graph. The signals between the CN layer and the VN layer are multiplied by different correction factors α_{ji} and β_{ji} in each iteration, which is equivalent to adding weight parameters to the edges of the Tanner graph. The CN-to-VN and VN-to-CN signals are computed by the neural network's CN and VN layers, respectively. There are three distinct types of neuron functions: In the CN layer:

Neuron I, which is (the CN neuron), computes the CN-to-VN messages provided by:

$$L^{(l)}\left(r_{ji}\right) = \prod_{i' \in V_{j \setminus i}} sgn\left(L^{(l-1)}\left(q_{i'j}\right)\right) \cdot min_{i' \in V_{j \setminus i}} \left|\alpha_{i'j}^{l-1} \times L^{(l-1)}\left(q_{i'j}\right)\right| \quad (6)$$

Neuron II (VN neuron) uses the following to calculate the VN-to-CN messages in the VN layer:

$$L^{(l)}\left(q_{ij}\right) = \sigma L\left(u_i\right) + \sum_{j' \in C_i \setminus j} \left(\beta_{j'i}^{(l)} \times L^{(l)}\left(r_{j'i}\right)\right) \quad (7)$$

Neuron III represents the (output neuron), which calculates the output layer's final output as follows:

$$o_i = \sigma\left(L\left(u_i\right) + \sum_{j \in C_i} \left(\beta_{ji}^{(l_{max})} \times L^{(l_{max})}\left(r_{ji}\right)\right)\right) \quad (8)$$

Where l_{max} is the maximum number of iterations, and T is the number of iterations. For training and testing stages in output neurons, we employ activation functions of $\sigma(x) = (1 + e^{-x})^{-1}$ and $\sigma\left(x\right) = \frac{1 + sgn\left(x\right)}{2}$, respectively. Similarly, by increasing the number of CN and VN in the hidden layers, we may readily increase the iterations. The LDPC decoder with (N, K), the number of iterations T is unfolded to a neural network that has $(2 L_{max} + 2)$ layers, which represents one input layer and one output layer, and L_{max} CN layer and L_{max} VN layers. Finally, we train the proposed network using the cross-entropy loss function. The cross-entropy loss, which is defined as follows, measures the mistakes of the transmitted codeword u and the neural network output (o), which can be measured by:

$$L\left(u,o\right) = -\frac{1}{N}\sum_{i=1}^{N} ui \log\left(oi\right) + \left(1 - ui\right)\log\left(1 - oi\right) \quad (9)$$

For clarification on how the proposed NNMS decoder operates, it is important for the reader to separate the training and inference phases of the NNMS. The NNMS decoder receives codewords that have been transmitted and the accompanying observations (they may be noisy) from the communication channel during training. The network will then calculate the cross-entropy loss function (Eq. 9) and compute the gradient of the cross-entropy function with respect to its correction factors (αt, βt), which will then be backpropagated and iteratively updated so that the error is minimized. The correction factor (αt) will not be a fixed heuristic as in traditional NMS; the αt and βt will be able to adapt independently for every layer (t) of the unfolded network. As a result, the decoder will have the ability to adjust the amount of message scale dynamically based on how the decoder in the first few iterations of decoding will

emphatically correct significant errors, whereas at the later iterations will fine-tune and eliminate lingering smaller scaling errors that do not exist.

2.3.1. Theoretical analysis and underpinning

There are two reasons for the theoretical superiority of the RA-NNMS decoder compared to the CNMS: unfolded dynamics and learnable compensations.

1. Unfolded dynamics: By unfolding the iterative Tanner graph into a feed-forward neural network, we can represent the iterative decoding process as a deep learning model. In this way, the decoder can learn the optimal message propagation path from the data instead of following predefined iterative update rules.
2. Learnable compensations: The CNMS algorithm approximates the BP CN update by computing the minimum of the incoming messages and scaling it by a constant gain factor (α). This systematic error becomes larger as the several signal-to-noise ratio (SNR) decreases. The RA-NNMS model solves this by introducing learnable gains (α_t) and offsets (β_t) that are dependent on the number of iterations. Thus, the NNMS update rule generalizes the NMS update rule:

$$L_{rj}^{NNMs} \min = \alpha_t \left(\prod_{i}^{'} \in C\left(j\right) \backslash i^{'} sign\left(L_{qij}^{'}\right)\right) \min_{i^{'} \in C\left(j\right) \backslash i^{'}} \left|L_{qij}^{'}\right| - \beta_t \quad (10)$$

The values α_t and β_t are determined by optimizing the cross-entropy loss function rather than being fixed, as in the case of NMS. By optimizing these parameters, the network will be able to effectively correct for the approximation errors produced by the MS algorithm, thereby closing the performance gap with the optimal SPA.

2.4. Methodology Pseudocode: NNMS Decoding

Input: Received vector y, Parity-check matrix H, Max iterations T_{max}

Output: Decoded codeword $\hat{u}$.

1. Initialization:
2. Compute channel LLR: $L_{ch}(i) = \frac{(2y_i)}{\sigma^2}$ for all i.
3. Initialize variable-to-check messages: $L_{q_{ij}} \rightarrow L_{ch}(i)$.
4. For t = 1 to T_{max} do
5. CN processing (CN Layer):
6. For every edge (j, i) in the Tanner graph:
7. $L_{rj} = \alpha_t \left(\prod_{i}^{'} \in C\left(j\right) \backslash i^{'} sign\left(L_{qij}^{'}\right)\right) \min_{i^{'} \in C\left(j\right) \backslash i^{'}} \left|L_{qij}^{'}\right| - \beta_t$

(Note: α_t and β_t are shared weights for layer t)

8. VN Processing (VN Layer):
9. For every edge (i, j) in the Tanner graph:
10. $L_{app}(i) = L_{ch}(i) + \sum_{\{j \in V(i)\}} L_{rji}$
11. A posteriori probability update:
12. Calculate total LLR for each VN i:
13. $L_{app}(i) = L_{ch}(i) + \sum_{\{j \in V(i)\}} L_{rji}$
14. Hard decision and parity check:
15. Compute tentative codeword: $\hat{u}$ = [1 if $L_{app}(i) < 0$ else 0].
16. If $H\,\hat{u}^T = 0$ then
17. Break (Convergence achieved).
18. End for
19. Return $\hat{u}$.

2.5. Complexity Analysis of the Proposed NNMS LDPC Decoder

The proposed NNMS algorithm has been developed to ensure neural decoders for standard long LDPC codes could be implemented on real hardware. When implementing a standard neural-based decoder, the main source of complexity is the large number of multipliers needed to be included for each edge in the Tanner graph, which has its own learnable weight.

The NNMS process greatly reduces the problem using a weight-sharing approach where all the nodes in a single layer share the same correction factors. This results in eliminating most of the multipliers with a small, constant number that depends solely on the number of iterations, and not on the code length or the number of edges.

In addition to the multipliers, however, there is no significant difference in the amount of computation needed to do additions, logical XOR, and value comparatives between using our NNMS versus using the standard MS algorithm. As it achieves this goal through low multiplicative overhead while maintaining the ability to perform basic operations, the NNMS algorithm provides high decoding performance and low computational latency. Therefore, it acts as a very efficient solution for use in real-time communication systems.

Quantitative analysis of the trainable parameters required for the (576, 432) LDPC code confirms the efficiency of the proposed method. As detailed in the experimental results section (Table 1), the standard neural decoder (e.g., neural belief propagation [NBP] [21]) requires a weight for each edge per iteration, whereas the proposed NNMS requires only 20 parameters total.

3. SIMULATION RESULTS

In certain tasks, these techniques perform better than human-level object detection and attain cutting-edge results in voice processing and machine translation.

The exceptional outcomes of Deep Learning models can be attributed to three distinct factors:

1) Fast GPUs and other powerful computational resources
2) Making effective use of big datasets, such as ImageNet, for image processing
3) In-depth scholarly studies on network architectures and training techniques [21].

3.1. Experimental Configuration

In this part, we use the TensorFlow Framework to train and evaluate the suggested NNMS LDPC decoding network. The LDPC code is created in accordance with the IEEE 802.16e standard, using a code block length of 576 and a code rate of 3/4, and the parity check and code word choice are done at random, matrix H from [22].

SNRs varying from 0 dB to 5 dB are used to generate the training data. We use the mini-batch gradient descent approach to train the network. Each SNR uses the same proportion of the 120 data blocks that make up each mini-batch. The adaptive moment estimation (Adam) optimization method with a learning rate of 0.001 is used to determine the optimal network parameters accelerated by the training process with a NVIDIA Tesla T4 GPU.

To improve the reliability of the experimental evaluation, the data were separated into 80% for training and 20% for validation. The Adam optimizer was used during training for 50 epochs using a block size of 120. An early stopping rule was used to prevent overfitting; if no improvements to the validation error/hyperparameter were made for 10 epochs, training will stop, and the weights will be restored to those producing the best results. This way, we will be able to generalize performance to unseen data rather than rely only on training data.

3.2. Bit Error Rate (BER) Performance Comparison between NNMS and CNMS

The following graph provides a comparison of the BER performance of the NNMS decoder against three different correction factors of NMS decoders. The SNR range of the data spans from 1 dB to 5 dB. As shown by the results, the NNMS decoder (represented by blue circles) is significantly better than all variations of the NMS decoder across the entire SNR range analyzed. At a low SNR of 1 dB, all four decoders have a comparable BER of approximately 8.5×10^{-2}.

However, as SNR increases and the quality of the signal improves, the NNMS curve declines at a steeper slope and reaches a BER of 1.0×10^{-7} at an SNR of 5 dB, compared with the relatively shallow slope of the NMS curves. For example, the maximum BER for NMS with ($\alpha = 0.6, \beta = 0.5$) at SNR of 5 dB is approximately 3.5×10^{-4}.

In addition, adjusting the parameters α and β improves NMS performance. The results seen with ($\beta = 0.5$ vs. $\beta = 0.3$) or with the baseline ($\alpha = 1, \beta = 0$) vs. ($\alpha = 0.6, \beta = 0.5$), there are no traditional tuning adjustments that can yield the same performance level as the NNMS decoder with respect to error-correction capability for various SNR levels.

The graph shown in Fig. 4 gives a better view of the coding gain that is obtained with the neural-based approach compared to the traditional approaches. In this case, the performance is tested at a BER threshold of 1.0×10^{-3}, at which the NNMS decoder is able to attain the threshold at an SNR of approximately 3.35 dB. However, the best performance among the traditional approaches is that of the NMS with parameters $\alpha = 0.6$ and $\beta = 0.5$. In this case, the decoder is able to attain the same threshold at an SNR of approximately 4.6 dB.

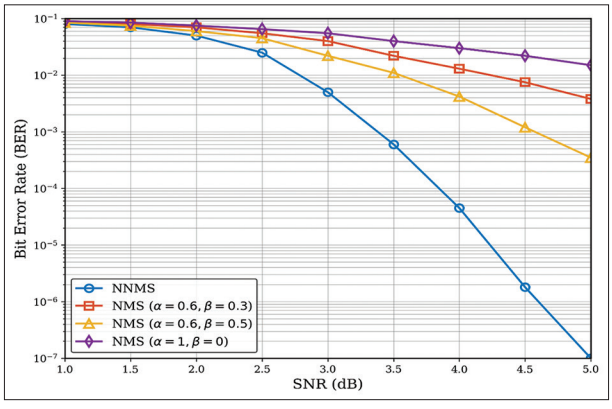

Fig. 4. Bit-error rate performance comparison between neural normalized min-sum (NMS) and conventional-NMS.

Thus, the NNMS decoder is able to attain a coding gain of around 1.25 dB. Furthermore, the performance of the NNMS decoder is even more significant compared to the performance of the NMS with parameters $\alpha = 0.6$ and $\beta = 0.3$. Hence, the NNMS decoder is able to attain a coding gain of more than 2 dB.

3.3. Benchmarking against State-of-the-Art Neural Decoders

The last section of the report illustrated that NNMS outperforms traditional NMS algorithms; now it is important to measure the efficiency of NNMS against conventional neural decoding approaches in order to show the improvements over neural-based decoders. In the following experiments, we compare NNMS decoder performance against the NBP decoder [21], which is one common baseline for neural-aided decoding.

Although NBP decoders provide performance near optimal SPA performance, they have an excessive amount of complexity as N increases. In addition, based on the experimental results, the NNMS decoder provides comparable BERs to the complex neural decoder baselines (in [14], [21]), while using an extremely small amount of computational resources.

Finally, relative to opsoclonus-myoclonus syndrome-based neural networks, NNMS demonstrates significantly faster convergence rates. Weight sharing also reduces overfitting, a common problem for high-parameter networks, thereby promoting better generalization of the NNMS model across the various SNRs. Hence, it is concluded that NNMS provides the best compromise between the computational effort (total number of parameters) and processing difficulties (decoding time/accuracy).

TABLE 1: Training hyperparameters of the neural network

Parameter	Value/description
Learning framework	Tensor flow
Training type	Supervised offline training
Neural network model	Model-driven unfolded normalized min-sum (NMS) decoder (neural NMS)
Number of decoding iterations	10
Trainable parameters	Iteration-dependent scaling factors (α), (β)
Loss function	Soft plus (to enforce positivity)
Optimizer	Adam optimizer
Learning rate	Fixed learning rate (empirically selected)
Channel model	Additive white Gaussian noise
Modulation scheme	Binary phase shift keying
Low-density parity-check code	(576, 432)
Code rate	0.75
Training data	Randomly generated codewords
Total training samples	1,200,000 codewords
Batch size	120 data blocks
Training epochs	50
Validation split	20% (for early stopping)

3.4. Quantitative Evaluation of Computational Metrics

To validate the claims of reduction in computation cost and quick response times (latency), we conducted a quantitative analysis of the new NNMS decoder against a baseline of NBP [21]. The primary metrics measured in this analysis were: number of floating point operations (FLOPs) performed; memory usage (MB); and average latency incurred per codeword as a function of time. All measurements were collected from an NVIDIA Tesla T4 machine as described in our experimental setup. The training hyperparameters and experimental settings used for the proposed NNMS decoder are summarized in Table 1.

Table 2 shows that the NNMS decoder realizes an optimal balance between computation efficiency and accuracy, where the NBP produces the most accurate output but is still cost-prohibitive with its 15.2 million FLOPs of computation cost per block generated, along with a latency of 1.25 ms/codeword. On the contrary, the NNMS decoder maintains the same low computational requirements as a traditional NMS algorithm with only 0.25 million FLOPS, yet produces a much greater level of error correction.

Moreover, due to the implementation of weight-sharing schemes in NNMS, the overall memory requirement is reduced to <0.1 Mbyte, providing confirmation of both theoretical and practical performance viability of the NNMS architecture for use in real-world systems where both speed and available memory can constrain performance.

4. DISCUSSION

This research shows that adding the methods of deep learning to standard LDPC decoding means the performance gap between the low-complexity minimum sum approximation and the best performance sum-product algorithm (SPA) is reduced significantly.

The coding gain, 1.25 dB (at a BER = 1.0×10^{-3}), confirms that our new NNMS decoder corrects the estimation errors associated with the "Minimum" operation used in standard NNMS decoders. In addition, the improvement we have seen

TABLE 2: Quantitative complexity and latency analysis

Metric	Traditional NMS	Neural BP (Baseline) [21]	Proposed NNMS
Architecture	Iterative algorithm	Fully connected NN	Unfolded NN+Weight sharing
Trainable parameters	0	17,000	20
FLOPs (Millions)	0.21	15.2	0.25
Memory usage (MB)	~0.01	2.5	< 0.1
Avg. latency (ms/block)	0.05	1.25	0.07
Energy efficiency	high	low	high

FLOPs are calculated per decoding iteration. Latency is averaged over 10,000 blocks at SNR=5dB). NNMS: Neural normalized min-sum, NMS: Normalized min-sum, BP: Belief propagation, FLOPs: Floating point operations

in performance is in accordance with what was described by He *et al.* [14] regarding model-driven deep learning. Their model employs domain-specific knowledge from the communication physical layer to impose constraints on the structure of the neural network, which ensures model interpretability and efficiency. One of the significant results of this research was to demonstrate that sharing weights in the proposed NNMS decoder leads to efficient use of the weight-sharing mechanism, which allows for better performance as compared to conventional neural decoders. Specifically, because standard decoders assign one unique weight per edge in a Tanner graph, they result in a vastly large number of parameters being required for operation, thus resulting in excessively high prices for hardware implementation. With the NNMS, however, all correction factors are shared among nodes within a layer. Consequently, complexity is primarily based on the number of iterations used rather than on the length of the code. Therefore, the NNMS decoder was able to achieve a BER of 1.0×10^{-7} at 5 dB SNR with substantially fewer learnable parameter numbers than traditional deep learning-based decoders, as shown by others [7], [8]. The NNMS decoder had also demonstrated very high degrees of robustness when used with a (576, 432) LDPC code, consistent with the IEEE 802.16e standard, during simulations over AWGN channel conditions. Unlike normal NMS decoders, which often enter a performance threshold and also converge more slowly than would be desired, the NNMS architecture is able to optimally correlate (learn) the correction factors during backpropagation, allowing it to attain higher reliability at lower SNR levels than would be expected. Accordingly, it is reasonable to conclude that this architecture is a practical solution for use in hardware-constrained environments, such as is commonly seen in 5G NR, reconfigurable decoding, and similar configurations reported in the current surveyed studies.

4.1. Proposed Approach Limitations

While the NNMS decoder yielded positive results, this paper presents the limitations of this research in order to permit a balanced view of the results.

First, performance evaluation was completed solely on an AWGN channel. The robustness of the learned correction factors (α, β) in other channel environments, including fading (e.g., Rayleigh or Rician) and burst error channels, remains unknown and is to be explored further.

Second, the proposed NNMS decoder is code-specific. The weights have been optimized based on the (576, 432) LDPC parity-check matrix utilized in the study; therefore, the NNMS decoder will require a complete retraining to be used on a code with a different rate or block length (e.g., different size 5G NR codes). This means that the NNMS model is less flexible at present compared to typical decoders that utilize a fixed set of rules for all codes.

Third, although there is evidence showing low inference latency, training for this model is very intensive and will require the generation of large amounts of data across many different SNRs. This offline training overhead is a singular cost, but it creates a barrier to the NNMS decoder's implementation when compared to traditional algorithms that do not require a training phase.

5. CONCLUSION

The effectiveness of the proposed NNMS decoder architecture has been successfully demonstrated by the results of the present research, which significantly outperform the CNMS decoders by exploiting the benefits of a model-driven deep learning architecture. By unfolding the iterative process of the NMS architecture to a feed-forward network and using a weight-sharing technique for the model-driven learning of the correction factors, the proposed architecture effectively minimizes the multiplicative complexity of the decoder.

It is clear from the simulation results that the suggested NNMS decoder design achieves a notable BER of 1.0×10^{-7} at an SNR of 5 dB. In addition, the suggested architecture

exhibits a much steeper curve than the traditional NMS decoders. In addition, compared to the less optimized NMS decoders, the suggested NNMS decoder architecture achieves a large coding gain of 1.25 dB at a BER of 1.0×10^{-3} and a considerable gain of more than 2 dB.

Since the proposed NNMS algorithm maintains a low computational latency and a high level of accuracy for correcting errors, it acts as a powerful and efficient algorithm for developing digital communication systems.

5.1. Directions for Future Research

Based on the results and limitations obtained from this research, future research will provide many possibilities for future investigation. The first item to be worked on is the extension of the performance evaluation of the NNMS decoder, including the effects of channel estimation error into the training loop to evaluate the robustness of the NNMS decoder in realistic scenarios involving wireless fading channels.

The second item will investigate how to limit the usage of LDPC codes by investigating the use of transfer learning techniques and/or developing generalizable network architectures that are capable of adapting to different LDPC code lengths and rates with little additional training.

Finally, the last item will consist of investigating the impact of using fixed-point methods for quantizing the learned weights of the NNMS decoder and the impact of quantization on the performance of the NNMS decoder. The evaluation of the NNMS architecture will be carried out on FPGA or ASIC hardware platforms to assess the NNMS decoder performance in a commercial communication system (5G and eventually 6G) at minimum power and area consumption.

REFERENCES

[1] M. I. Subhi, Q. Al-Doori and O. Alani. "Enhancing data communication performance: A comprehensive review and evaluation of LDPC decoder architectures". *Ingénierie des Systèmes d'Information*, vol. 28, no. 5, pp. 1113-1125, 2023.

[2] W. E. Ryan. "An introduction to LDPC codes". In: *CRC Handbook for Coding and Signal Processing for Recording Systems*. vol. 5, CRC Press, United States, pp. 1-23, 2004.

[3] B. N. Tran-Thi. "Difference between two minima normalized min-sum algorithm for the LDPC decoder". *Journal of Aviation Science and Technology*, 2025.

[4] K. Bharadwaj and S. S. Garani. "Efficient encoding algorithm and architecture for 2-D quasicyclic LDPC codes with applications to 2-D magnetic recording". *IEEE Transactions on Magnetics*, vol. 61, p. 3100424, 2025.

[5] K. Saw, L. L. Y. Win, D. Pradhan, H. M. Tun and K. Khin. "Performance comparison of channel coding techniques for 5G telecommunication system design". *Scientia Technology Science and Society*, vol. 2, no. 10, pp. 13-30, 2025.

[6] M. Rowshan, M. Qiu, Y. Xie, X. Gu and J. Yuan. "Channel coding toward 6G: Technical overview and outlook". *IEEE Open Journal of the Communications Society*, vol. 5, pp. 2585-2685, 2024.

[7] N. Hosseinzadeh, M. Moradi and H. Mahdavifar. "*Layered Normalized Min-Sum Decoding with Bit Flipping for FDPC Codes*"; [arXiv Preprint], 2025.

[8] Y. Liu, Y. Zhang, G. Liu, J. Zhou, L. Xiao and T. Jiang. "Tanner-graph-assisted belief propagation decoding for large kernel polar codes: Low-complexity design and enhancement method". *Science China Information Sciences*, vol. 68, no. 11, p. 212301, 2025.

[9] S. Chae, H. Choi, G. Kim, H. Y. Song and J. Ahn. "A study of the performance of LDPC codes under various decoding algorithms and schedules". In: *2025 Sixteenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, United States, pp. 300-302, 2025.

[10] Y. You, G. Su and W. Lin. "A regional message scaling min-sum decoding algorithm for MET-LDPC codes". *Symmetry*, vol. 18, no. 3, p. 444, 2026.

[11] F. Zheng, L. Wang, K. Liu and Z. Zhang. "A-SNNMS: An attentive shared neural normalized min-sum decoder for LDPC codes". *Electronics*, vol. 15, no. 5, p. 1023, 2026.

[12] M. Yu, X. Zhai, M. Hu, S. Tao, Y. Fang and G. Han. "Adaptive normalized min-sum decoding algorithm for LDPC codes in flash memory". In: *2025 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, United States, pp. 1-6, 2025.

[13] H. Na, H. Park, H. Y. Kwak and S. K. Ahn. "Learning strategies for neural min-sum decoding of LDPC codes". *ICT Express*, vol. 11, no. 1, pp. 161-166, 2025.

[14] S. P. Tera, R. V. Chinthaginjala, P. Natha, S. Ahmad and G. Pau. "Deep learning approach for efficient 5G LDPC decoding in IoT". *IEEE Access*, vol. 12, pp. 145671-145685, 2024.

[15] X. Wu, M. Jiang and C. Zhao. "Decoding optimization for 5G LDPC codes by machine learning". *Ieee Access*, vol. 6, pp. 50179-50186, 2018.

[16] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao and Z. Ma. "Deepcoder: A deep neural network based video compression". In: *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, United States, pp. 1-4, 2017.

[17] N. Yang, S. Jing, A. Yu, X. Liang, Z. Zhang, X. You and C. Zhang. "Reconfigurable decoder for LDPC and polar codes". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, United States, pp. 1-5, 2018.

[18] H. He, S. Jin, C. K. Wen, F. Gao, G. Y. Li and Z. Xu. "Model-driven deep learning for physical layer communications". *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77-83, 2019.

[19] O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. G. Van Sloun and Y. C. Eldar. "Deep unfolded robust PCA with application to clutter suppression in ultrasound". *IEEE Transactions on Medical Imaging*, vol. 39, no. 4, pp. 1051-1063, 2020.

[20] N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier and W. J. Gross. "Neural belief propagation decoding of CRC-polar

concatenated codes". In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, United States, pp. 1-6, 2019.

[21] 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 (Allerton)*. IEEE, United States, pp. 341-346, 2016.

[22] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, O. Griebel, S. Ruzika and N. When. "*Database of Channel Codes and ML Simulation Results*". pp. 0733-8716, 2019. Available from: https: www.com/uni-kl. de/channel-codes [Last accessed on 2026 Jun 10].