Parallel Non-Iterative Cascaded Feedforward Neural Network Decoder for Low-Density Parity-Check Codes

Harshawardhan P. Ahire, *Member, IAENG*, Sushama R. Wagh, *Member, IAENG*

***Abstract*—Low-density parity-check (LDPC) codes are widely used in modern systems because they are highly effective for error correction, nearing the Shannon limit performance across diverse communication channels. However, choosing an appropriate decoder for LDPC is crucial for accurate information retrieval. Traditional decoders, such as the message-passing sum-product algorithm (MP-SPA), message-passing belief propagation (MP-BP) and its variants, often suffer from computational complexity, error floors, and latency owing to their probabilistic and iterative nature. This experiment utilizes a cascaded feedforward neural network (CFNN) as a non-iterative decoder for quasi-cyclic (QC) LDPC codes. The CFNN completed 376 iterations within 43 minutes and 21 seconds, achieving 41.9% performance with a 3.62^{-02} gradient over six validation checks. The bit error rate (BER) of the CFNN decoder for the QC-LDPC improved by $10^{-0.24}$ compared to conventional decoders at a signal-to-noise ratio (SNR) of 3, with the CFNN reaching a BER of $10^{-4.5}$ at 5 dB SNR versus $10^{-3.6}$ for conventional decoders. The results show that the CFNN decoder excels. Overall, the CFNN proved to be an effective non-iterative alternative, overcoming the limitations of traditional decoders, such as suboptimal performance and error floors. The enhanced BER performance at various SNR levels demonstrates the efficacy of the CFNN decoder in improving decoding accuracy and reliability in communication systems.**

***Index Terms*—Belief Propagation, Low-Density Parity-Check Code, LDPC Decoder, Neural Network, Quasi-Cyclic, Regular LDPC.**

I. Introduction

GALLAGER [1] introduced low-density parity-check (LDPC) codes based on linear block codes in 1962. In 1999, Mackay rediscovered LDPC codes, highlighting their capacity-approaching performance, as characterized by Shannon [2], [3]. Tanner [4] provided a graphical representation of sparse parity check matrices (PCM), emphasizing their role in reducing computational complexity during LDPC decoding. Juane et al. [5] discussed the widespread application of LDPC codes in deep-space communication, image authentication, mobile communication, wireless sensor networks, and various wireless standards owing to their efficient decoding algorithms and excellent performance. Saurabh [6] noted that LDPC codes offer high throughput, parallelizable hardware implementation, low decoding latency, and near Shannon capacity performance, making them essential for channel coding.

Manuscript received February 7, 2024; revised December 22, 2024.

H. P. Ahire is a PhD candidate of Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai, 400019, INDIA. (e-mail: hpahire_p17@el.vjti.ac.in)

S. R. Wagh is a professor of Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai, 400019, INDIA. (e-mail: srwagh@ee.vjti.ac.in)

TABLE I
Abbreviations Used

Abbreviation	Full form
LDPC	Low-Density Parity-Check
PCM	Parity-Check Matrix
URLLC	Ultra Reliable Low Latency Communications
AWGN	Additive White Gaussian Noise
CFNN	Cascaded Feedforward Neural Network
MSNN	Multilayer Shallow Neural Network
MN	Mackay-Neal
QC	Quasi-Cyclic
GF	Galois Field
LSTM	Long Short-Term Memory
MP-SPA	Message Passing Sum-Product Algorithm
MP-BP	Message Passing Belief Propagation
LBP	Linear Belief Propagation
OMSBP	Offset Min-Sum Belief Propagation
SNR	Signal-to-Noise Ratio
LLR	Log Likelihood Ratio
SCG	Scaled Conjugate Gradient
MSE	Mean Square Error

Zhou [7] found that the quasi-cyclic (QC) method for constructing PCM reduces the memory requirements and computational complexity. Khodaiemehr [8] found that the circulant permutation of the identity matrix enhances the decoder's bit error rate (BER) by obtaining PCM with higher Galois field (GF) orders. Kou [9] observed that constructing LDPC codes using algebraic methods based on finite analytic geometries results in a robust error performance near the Shannon limit, providing viable alternatives to turbo codes. Patil [10] noted that in 5G wireless communications, LDPC and Polar codes replace Turbo Codes and Tail Biting Convolution Codes because of their superior error-correction capabilities and high coding gain, which are crucial for the mobile Internet and IoT. Jiang et al. [11] introduced a flexible coding approach (FLCA) for QC-LDPC codes in 5G new radio (NR), optimizing shortening patterns for various code rates and modulation schemes and significantly improving performance over current LDPC codes. Ro [12] enhanced the reliability of weak variable nodes in protograph-based raptor-like (PBRL) LDPC codes by adding edges to the protograph, achieving low-error floors for ultra-reliable low-latency communications (URLLC), demonstrated on 5G NR-LDPC codes. Golmohammadi [13] proposed a novel windowed decoding scheme for concatenated spatially coupled LDPC codes in joint source-channel coding, showing improved performance over existing schemes.

Kim [14] introduced a new class of irregular LDPC codes optimized for finite-block-length applications. These codes feature efficient encoding and a simple rate-compatible puncturing structure and outperform optimized irregular LDPC and

(extended) irregular repeat-accumulate codes, particularly at high puncturing rates. They also enhanced the throughput in incremental redundancy hybrid automatic repeat request (IR-HARQ) systems over time-varying channels. Cole [15] developed a three-step procedure to efficiently determine the low-bit-error rate performance curve for a wide range of LDPC codes of moderate length, allowing for analysis and design without exhaustive error event searches or Monte Carlo simulations. Dolecek [16] created a deterministic method to predict error floors in LDPC codes using a high signal-to-noise ratio (SNR) asymptotically applied to absorbing sets within structured LDPC codes, significantly reducing computational requirements and extending error probability predictions to 10^{-30}.

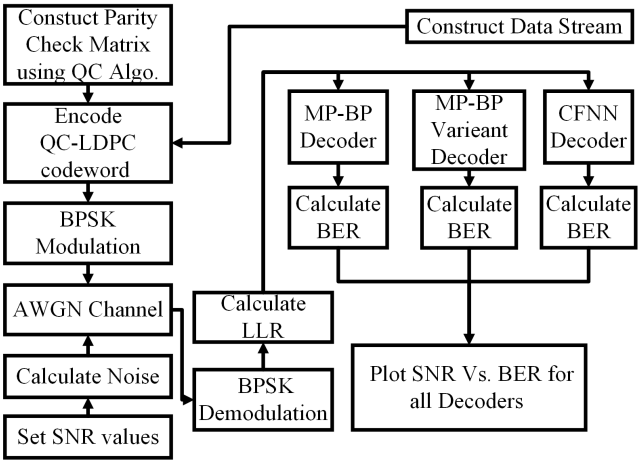

Fig. 1. Decoding scheme of LDPC code using MPBP, Variants and CFNN

Vatta [17] conducted a detailed mathematical analysis of belief-propagation decoding for LDPC codes on memory-less binary-input Additive White Gaussian Noise (AWGN) channels, establishing stricter bounds for the function $\phi(x)$ defined by Chung et al. [18]. This approach enabled a more precise assessment of the long-term performance of sum-product decoding for LDPC codes using the Gaussian approximation. Milenkovic [19] employed the large deviation theory and statistical techniques to assess asymptotic normalized average distributions of trapping and stopping sets in various LDPC code ensembles, including random, regular, and irregular binary types. These distributions are vital for determining the error floor performance curve and extending it to broader structural elements such as sub-codes and minimal codewords. Chen [20] introduced a method for generating irregular LDPC codes through quasi-cyclic extension. This approach produced codes with a low error floor in high SNR regions, minimized undetected errors, and retained efficient encodability. Lee [21] introduced a design technique using a trellis search to create effective LDPC codes with low code rates, improve the cycle distribution within PCM entries and outperform traditional greedy design algorithms, as demonstrated by simulations. Smith [22] developed a numerical method to minimize the decoding complexity in long-block-length irregular LDPC codes and found that complexity-optimized codes outperform threshold-optimized codes for long block lengths when the decoding complexity is constrained.

Zhang et al. [23] investigated LSTM networks in a neural network-based decoder to mitigate significant decoding delays and performance declines under non-Gaussian noise in turbo decoding algorithms. Their methodology demonstrated improved performance and reduced computational complexity compared with conventional approaches. Condo [24] demonstrated that an NoC-based strategy for multi-standard decoders in wireless receivers leads to higher throughput and similar or reduced area occupancy compared to previous implementations, achieving over 70 Mb/s throughput with a 3.17 mm^2 area on 90 nm CMOS technology. Chu [25] proposed a NOLD algorithm for LDPC codes, enhancing decoding parameters via a combined approach using a NOLD decoder and a neural network, resulting in superior performance compared to traditional decoders across various channel conditions. Guangwen [26] introduced a method for determining weights in belief propagation decoding variants of LDPC codes as trainable parameters within a deep learning framework, focusing on high-quality training data, the relationship between training loss and decoding metrics, and reducing decoding complexity by minimizing the trainable parameters. Extensive simulations validated the efficacy of this method. Ma [27] presented a Quasi-Resnet architecture for BP decoding of LDPC codes, enhancing performance by transmitting reliable messages between iterations and adjusting shortcut connection weights via the error backpropagation algorithm, as corroborated by the simulation results.

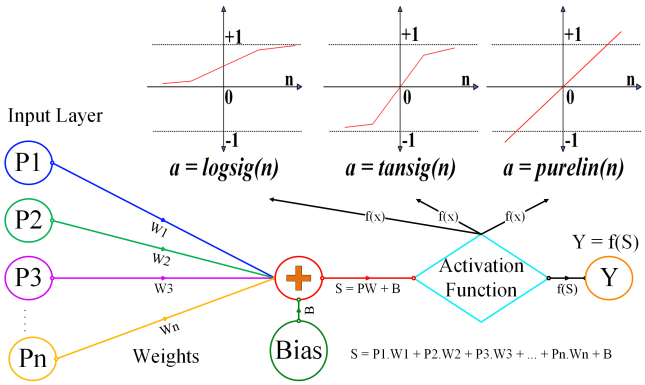

Fig. 2. Construction of a Perceptron with activation functions

Aggerwal [28] explained that multilayer shallow neural networks (MSNN) mimic biological neural networks and function as nonlinear statistical data models. MSNNs are effective and efficient for pattern recognition applications. Das et al. [29] described a pattern as a set of images, elements, articles, facts, bearings, or sequences of 0 and 1. Chang [30] provided a methodology to determine when the hidden layers of multilayer neural networks (MNN) share topological similarity, established criteria and examined the feasibility of replacing an MNN with fewer layers. In the cascaded feedforward neural network (CFNN) architecture, the internal component is the perceptron. For a perceptron with n inputs x_i and weights w_i, the output y is given by $y = \sum_{i=1}^{n} x_i \times w_i + BIAS$. The $BIAS$ is a learnable parameter for decision boundaries and model flexibility. Han [31] stated that the perceptron output is typically passed through an activation function, often a sigmoid function. Guo [32] noted that activation functions such as $tansig$

and $logsig$, followed by a threshold to determine the hard decision value of each bit. Supervised learning divides the decoding process into two distinct parts. The scaled conjugate gradient (SCG) learning algorithm, known for its speed, was used for the training. The inputs to the CFNN match the number of bits in the transmitted message with $tansig$ as the activation function. The contribution of the CFNN decoder to the existing decoding system is as follows.

1) **One-Shot Design:** The CFNN decoder provides output once the input is applied.
2) **Low Latency:** Because the process is non-iterative, less time required.
3) **Less Complex:** The decoding process becomes simple owing to the one-shot design. The only complexity is during the training.
4) **No Need to Transmit PCM:** After tuning the weights w_n and $BIAS$ during training, recreating the original message is unnecessary.

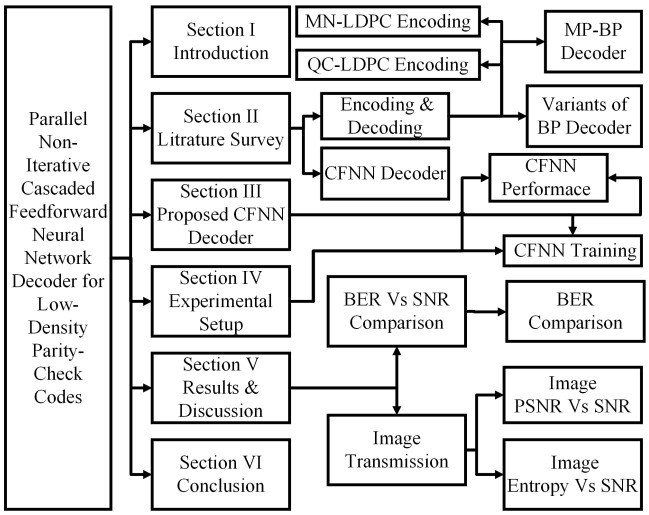

Fig. 3. Organization of Paper

The remainder of this study is structured as follows. A literature survey of encoding and decoding methods Quasi-Cyclic and Mackay-Neal is anticipated with the MP decoder using the SPA (MP-SPA) described in Section II. The proposed methodology for decoding LDPC codes using a CFNN in addition to CFNN training for decoding is discussed in Section III, and the experimental setup is discussed in Section IV. The testing procedure and arising empirical results are presented in Section V. Finally, concluding remarks are accorded in Section VI. The paper layout is shown in Fig. 3 and Table I list of the abbreviations used in this manuscript.

II. Literature Survey for LDPC Encoding and Decoding

To ensure error-free transmission of a binary message stream, an LDPC is employed as a channel encoder to incorporate redundant bits. The resulting encoded bit stream undergoes binary phase-shift keying (BPSK) modulation before transmitting through an AWGN channel. At the receiver, the decoder scheme calculates the log-likelihood ratio (LLR) for the demodulated signal, indicating that the probability of a specific bit is one. The LLR values were utilized to decode the LDPC code through various methods, including MPBP, its variants, and the CFNN decoder scheme. Ultimately, the original binary message stream was reconstructed at the receiver.

A. Encoding Techniques for LDPC

MacKay [33] presented empirical evidence demonstrating that Gallager LDPC codes surpass conventional convolutional and concatenated codes in terms of performance on Gaussian channels, achieving near-turbo code efficiency that approaches the Shannon limit. The investigation revealed that reducing the column weight, particularly during construction, significantly enhanced performance, nearly attaining the Shannon limit comparable to turbo codes. The increased block lengths yielded superior results, and codes with rates between $\frac{1}{2}$ and $\frac{1}{3}$ yielded optimal $\frac{Eb}{N0}$ values. Tanner [34] introduced algebraically structured QC-LDPC codes and their convolutional variants, demonstrating practical graph-based iterative decoding and the superior performance of LDPC convolutional codes over QC codes while also providing theoretical limits on girth and minimum distance based on their algebraic structure.

1) Mackay-Neal LDPC Encoder: Although the Mackay Neal (MN) method generates sparse code, it lacks a structured approach to code construction. The PCM construction technique determines the parity check bits by selecting the transmitted and source block lengths, where $M = N - K$. A minimum column weight of three is set to construct an $M \times N$ matrix. Random ones are inserted into each column to achieve balanced row weights. This ensures that the number of ones per row falls within the ratio range of $\frac{N}{M}$ to $\frac{N}{M}$. This method generates regular MN codes. Gaussian elimination was applied to modify the columns, creating a Systematic PC matrix pattern $H = (P|I_M)$. For odd t values, the LU decomposition technique was employed to address the potential matrix independence issues in obtaining the PCM. The LDPC codeword was constructed using the Generator Matrix (GM) $G = (I_M|P^T)$.

2) Quasi-Cyclic LDPC Encoder: Authentic LDPC code characteristics are obtained by fulfilling the generalized construction requirements, resulting in a diverse range of regular QC-LDPC codes. This flexibility applies to Convolutional and QC codes, with QC block codes possessing convolutional representation. The PCM derived from the convolutional depiction maintains the graph structure of genuine QC-LDPC codes, enabling efficient decoding through the MP algorithm. Convolutional codes offer several advantages in various applications. For short to moderate block lengths, irregular LDPC codes under perform algebraically structured QC-LDPC codes; however, they excel with longer blocks. An irregular LDPC code with list weight $w_c \geq 3$ is asymptotically good [2]. In structured QC-LDPC codes, large block dimensions and high code rates are achieved using circulant matrices in GF(m), where (m) is a prime number. The recurring symbol associations are generated using GF(m) nonzero elements, with elements x and y having argument positions $o(x) = p$ and $o(y) = q$, respectively. The GF(m) circulant matrix A is constructed by employing size $p \times q$

with the $(c,d)^{th}$ element $A_{p,q} = x^d y^c$ as follows:

$$A = \begin{bmatrix} 1 & x & x^2 & \cdots & x^{q-1} \\ y & xy & x^2y & \cdots & x^{q-1}y \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y^{p-1} & xy & xy^{p-1} & \cdots & x^{q-1}y^{p-1} \end{bmatrix} \quad (1)$$

Here, $0 \leq c \leq p-1$ and $0 \leq d \leq q-1$ as demonstrated in (2) PCM is derived from $p \times q$ cyclically shifted versions of the identity matrix in the act of a sub matrix. The Parity-Check Matrix was adopted for the architecture of the LDPC code.

$$H = \begin{bmatrix} I_1 & I_x & I_{x^2} & \cdots & I_{x^{q-1}} \\ I_y & I_{xy} & I_{x^2y} & \cdots & I_{x^{q-1}y} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_{y^{p-1}} & I_{xy^{p-1}} & I_{x^2y^{p-1}} & \cdots & I_{x^{q-1}y^{p-1}} \end{bmatrix} \quad (2)$$

The PCM is seized by cyclically shifting the left rows of the identity matrix corresponding to the elements of the circulant sub-matrix at positions (c,d). The size of the resulting dual PCM is $(p \cdot e \times q \cdot e)$. Additionally, the associated code rate is given by $(R = 1 - \frac{p}{q})$. The estimates of those in the individual columns designated via p along with individual rows will estimate q 1's and so forth, illustrating that the binding estimate for H is (p, q). Furthermore, the encoded binary sequence of the message is converted to non-return to zero (NRZ) according to (3). BPSK then modulates it. The BPSK signal was transmitted over an AWGN Channel with varying SNR.

$$NRZ(x) = \begin{cases} 1, & E(x) = 0 \\ -1, & E(x) = 1 \end{cases} \quad (3)$$

$$LLR(x) = \log \frac{P_{x=0}}{P_{x=1}} \quad (4)$$

The transmitted signal was received at the receiving end. LLR values were acquired, followed by BPSK demodulation using (4), which was fed to the MPBP decoders and CFNN (proposed algorithm) for decoding.

Algorithm 1: PCM using QC Construction

Result: Structured PCM for QC-LDPC for encoding
Prime numbers x and y from set GF($2^m - 1$);
multiplicative order p = o(x) and q = o(y);
construct an Identity matrix I_e with size $e = 2^m - 1$;
for *c= 0; c = p; c++* **do**
  for *d =0; d = q; d++* **do**
    $A_(p,q) = x^d y^c$;
  end
end
for *c = 0; c = p; c++* **do**
  for *d = 0; d = p; d++* **do**
    P($c \times e + 1$ to $p \times e$, $d \times e + 1$ to $q \times e$) = Cyclically sifted I_e;
  end
end
Construct PCM $H = (PI_M)$ and GM $G = (I_M|P^T)$;

3) Decoding with MPBP: In probabilistic graphical models, MPBP serves as a robust algorithm for inference and optimization. This technique operates on factor graphs that illustrate the factorization of a joint probability distribution. The fundamental process of MPBP involves iterative exchange of messages (m) between nodes. The message $(m_{v_i \to c_j})$ sent from the variable node V_i to check node C_j is updated using (5).

$$m_{v_i \to c_j} = LLR_{x_i} + \sum_{c_j \in N(v_i) \to c_i} m_{c_k \to v_i} \quad (5)$$

where, $N_{(v_i)}$ is the set of check nodes connected to (v_i). Each check node (c_j) sends a message $(m_{c_j \to v_i})$ to each connected variable node (v_i) by using (6)

$$m_{c_j \to v_i} = 2\tanh^{-1}\left(\prod_{v_k \in N(c_j) \to v_i} \tanh\left(\frac{m_{vk} \to c_j}{2}\right)\right) \quad (6)$$

where $(N(c_j))$ denotes the set of variable nodes connected to (c_j).
After a predefined number of iterations or upon convergence, the belief (updated LLR) for each variable node (v_i) was computed as mentioned in (7).

$$LL\acute{R}_{(v_i)} = LLR_{(x_i)} + \sum_{c_j \in N(v_i)} m_{cj} \to v_i \quad (7)$$

The final decision for each bit is made based on the sign of $(LL\acute{R}(v_i))$ according to (8).

$$\hat{x_i} = \begin{cases} 0, & if LL\acute{R}(v_i) \geq 0 \\ 1, & if LL\acute{R}(v_i) < 0 \end{cases} \quad (8)$$

The MPBP demonstrates efficacy in large-scale applications, approaching the Shannon limit for substantial block lengths. Although MPBP decoding exhibits superior performance for LDPC code, it has certain limitations. The BP algorithm presupposes a tree-like Tanner graph structure. However, LDPC codes frequently contain cycles, which may result in erroneous solutions or failure to converge within a reasonable number of iterations, particularly for high-density or sub-optimally designed codes. MPBP decoding may manifest as an error floor phenomenon, wherein the error rate ceases to decrease exponentially as the SNR increases, which is often attributable to trapping sets or small cycles in the graph. In high-speed applications, the computational complexity of $tanh$ and $tanh$ inverse operations in check-node update can become a computational bottleneck, even with approximations. The BP decoding performance may deteriorate if the channel conditions deviate significantly from the assumed model, especially in non-Gaussian-noise environments. Implementing efficient BP decoders in hardware (such as FPGA or ASIC) presents challenges owing to the requirements for parallel processing and intricate memory access patterns. The storage and updating of messages for large LDPC codes necessitate substantial memory, creating constraints in resource-limited settings. BP decoding is less effective for short block lengths, where the performance gap compared to maximum-likelihood decoding is more pronounced. Errors in the initial LLR or during message passing can propagate through the network, leading to incorrect decoding.

Algorithm 2: Decoding of Received LDPC using MPBP Decoder

Result: Decoded Message using MPBP Decoding
Demodulated Signal received x_i;
PCM H, number of iterations itr SNR in dB;
Calculate LLR using $LLR_{(x_i)} = 4 \times x_i \times SNR$;
for *i = 1; OR*$(i = itr, syndrome = 0); i++$ **do**
  Calculate $m_{c_j \to v_i}$ using (5);
  Calculate $m_{c_j \to v_i}$ using (6);
  Calculate $\hat{LLR}_{(v_i)}$ using (7);
  Calculate $\hat{x_i}$ hard decision using (8);
  Calculate syndrome using $syndrome_i = x_i \times H$;
  if *OR*$(i = itr, syndrome = 0)$ **then**
    Decoded Message = $\hat{xi}$;
    Stop the Process;
  end
  else
    $LLR_i = \hat{x_i}$;
  end
end

4) Min-Sum BP: Liang [35] investigated a modified algorithm, the min-sum algorithm, to decrease the computational complexity while maintaining the essence of the more complex MP-SPA algorithm. The initialization process followed the same procedure, initializing messages similar to MP-SPA. Unlike the MP-SPA, which computes the exact product, the min-sum algorithm employs a minimum operation to approximate the product.

- Variable nodes update their messages based on the minimum values they receive.
- Ensure that nodes communicate efficiently by minimizing the number of incoming messages required, considering both positive and negative signals.

The decision was contingent on the final values following a predetermined number of iterations.

5) Normalized Min-Sum BP: The study conducted by Wang [36] revealed that the optimization approach elevated the efficiency of the min-sum BP algorithm. The initialization and message-passing processes are similar to the min-sum process, with the added step of applying a normalization factor to the messages. After calculating the minimum values, the normalization factor (usually less than one) scales the messages to better approximate the original SPA messages. The final values determine the decisions in the same manner as the min sum.

6) Offset Min-Sum BP: Lugosch [37] explored the min-sum BP algorithm which has several advantages over its forerunners. One of these benefits is the similarity of its initialization and message-passing procedures with those of the min-sum algorithm. However, a critical distinction is that the offset is subtracted from the minimum value before the message is passed. This adjustment effectively reduced the bias inherent in the min-sum approximation. Therefore, decisions were made using the adjusted final values.

7) Layered BP: Maammar [38] developed an enhanced variant of the BP algorithm to achieve faster convergence. The initialization procedure for this variant adhered to the standard MPBP guidelines. In contrast to updating all variable nodes and then all check nodes simultaneously, this method updates subsets of nodes sequentially, in a layered manner. Consequently, the information propagates more swiftly, leading to a faster algorithm convergence. In fewer iterations, the variable node values were determined.

III. Proposed CFNN Decoder

Decoding the LDPC code by adopting the CFNN as a decoder is a novel idea. Several researchers have proposed using machine learning (ML) or deep learning (DL) algorithms to decode coding techniques other than LDPC codes. The major problems faced by existing LDPC decoders are latency and computational complexity, which make them difficult to implement or non-implementable.

Jason [39] utilized an artificial neural network (ANN) based on an information processing model inspired by the biological nervous system. The ANN feed-forward (FF) method, which links the input layer to the output layer via multiple intermediate neuron layers, is widely applied across various neuron compositions. Researchers have concentrated on ANN-related issues such as the extension, characterization, and development of mathematical expressions in neural network methods.

A CFNN Decoder was proposed for LDPC codes, providing good performance, less decoding time, and lower computational complexity once the CFNN was trained. The only constraint in the CFNN decoder is the message length, which can be increased by cascading the CFNN decoders. The decoding scheme to decode the LDPC using the CFNN decoder in Fig. 1. The decoder design involves two steps: training the CFNN and validating the network. Subsequently, the CFNN used for the analysis using a trained CFNN for decoding.

The LDPC code decoding process employs a CFNN to optimize the network performance. The input set comprising the LLR values from the LDPC code was used to train the CFNN. These LLR values, derived from the demodulated received signals, are converted into a binary set of symbols as the target set. Boutillon et al. [40] explored the λ-min decoding technique and achieved a BER of 0.9 dB. Maximilian [41] demonstrated a BER of 0.15 dB for LDPC decoders, whereas Dinesh [42] showed a BER of 1.5 dB, achieving 1.3 dB BER with a non-Iterative LDPC code.

A. Training of CFNN

The Training data set consists of a binary equivalent of length m, where m is the length of the message. These non-binary ‘numbers were converted into a binary message stream of size m. This binary message stream is encoded into an LDPC code word using the QC-LDPC generator matrix. The code word was then modulated using the BPSK modulation technique. The BPSK-NRZ signal was transmitted over the AWGN channel. During the transmission, the SNR varied. The received signal was demodulated on the receiving side and the LLR values were calculated using (4). An input data set and target data set were formed.

These input and target data set were used to train the CFNN. This forms an input, making the CFNN adjust the weights and biases after training for the target defined by the binary values. The construction of the CFNN is illustrated in Fig. 2, which also shows the activation functions used.

Algorithm 3: Training CFNN

Result: CFNN with updated weights and biases
Construct a Training data set consisting of LLR and Corresponding binary data values;
Decide the number of Hidden Layer Neurons and Learning Rate;
Initial values of weight = 1 and bias = 0;
Construct the cascaded feedforward neural network with input value;
Initialize the parameters of the cascaded feedforward neural network;
Feed the Training set to CFNN;
while *Error is Greater than Mean Square Error (MSE)* **do**
  Train the CFNN for the Training set and Target Set;
  Get the error between the Target set and the obtained output;
end
Test the output for known data from the Training Set and Target Set;
Validate for random data not from the Training Set;
Save the network;

B. Performance Evaluation for CFNN Decoder

In this experiment, the pattern was set to 0 and 1, arranged in a specific pattern. For Pattern Recognition, computation from MPBP, its variants, and the CFNN was used. Considering the facts related to MPBP, it is clear that the hyperbolic tan is used to calculate the LLR for extrinsic probability [43]. The log of all values was calculated to obtain the new LLR for the extrinsic probability. The syndrome is calculated after making a hard decision regarding extrinsic LLR. If the syndrome is zero, then decoding is performed properly and decoding is stopped; otherwise, the process is repeated.
The architecture of the CFNN used in this experiment is illustrated in Fig. 2. For empirical aspiration, three hidden layers are utilized to indicate along with one input and one output layer. The size of the input layer depends on the code word length and the number of outputs depends on the number of rows in the PCM. After decoding, the error in the individual bit and separate block is calculated, which provides the BER for the individual SNR. The BER was calculated for all SNR values and plotted against the SNR values.
The performance of the CFNN decoder for decoding the LDPC code was checked. A binary message stream of random 0 and 1 of length equal to m bits is encoded to the QC-LDPC. Subsequently, it was transmitted over the AWGN channel using BPSK modulation techniques. The noise level was determined using the SNR values in decibels. The LLR was calculated for each bit using the received code after demodulation, and these LLR values were fed to the CFNN for decoding.

C. Decoding of LDPC codes Using CFNN

The soft signal is defined from the AWGN channel and received signal. The LDPC code was encoded using QC-LDPC utilizing a random value from a certain symbol set.

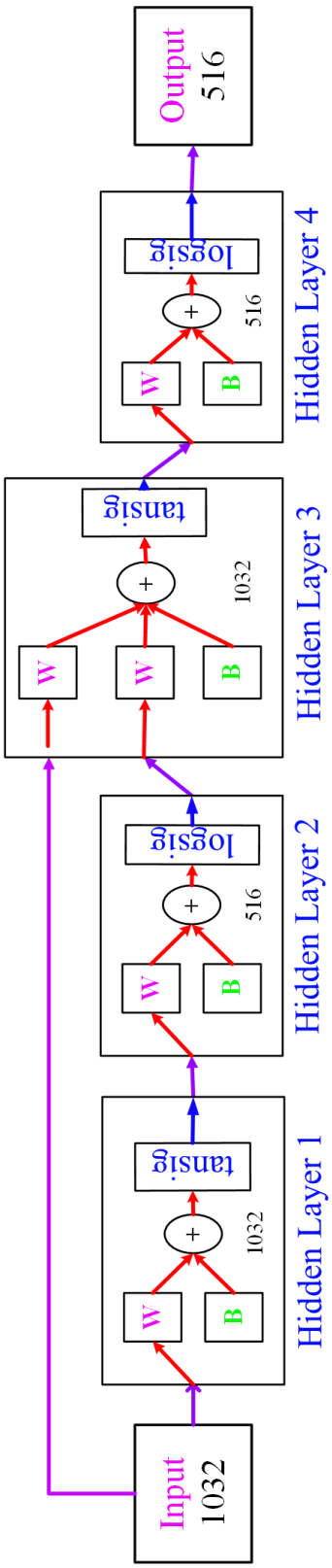

Fig. 4. Construction of cascaded feedforward neural network

The variation in the code rate considered in this study, that is, the channel SNR, varied for the AWGN channel. In the decoding process, random inputs were considered with the SNR value variable. For these SNR values, the BER variable with the AWGN channel is evaluated and used in this study. One layer of the CFNN is illustrated in Fig. 4, which also shows the activation functions used. The CFNN is

Algorithm 4: Performance of CFNN Decoder

```
Result: Decoded Message and BER for SNR
The number of frames;
Load the trained CFNN network;
Calculate the Upper Bound for each SNR;
for i = 1; i = Numbers of SNR; i++ do
    for j = 1; j = Number of Frames; j++ do
        Create a random binary stream;
        Encode LDPC codeword using Algorithm 1;
        Construct BPSK modulated signal and
         Transmit over AWGN Channel;
        Decode the received LLR using trained
         CFNN, MPBP and its variants;
        Calculate BER;
    end
end
Plot the SNR Vs. BER plot for the Upper Bound,
 CFNN, MPBP and its variants;
```

constructed with seven hidden layers inspired by the Tanner graph; the hidden layer 2, 4, 6, and output have nodes equal to the number of check nodes, that is, similar to the number of message bits with the **logsig** activation function. Hidden node 1,3,5,7, and the input consists of nodes equal to the number of variable nodes with the activation function **tansig**, which is, equivalent to the code word length. The last hidden layer carried nodes equal to the number of check nodes with a **tansig** activation function. The output state has a **max-pooling** activation, giving the probability of the bit to be 1. The inputs are added to hidden layers 1, 3, 5, 7 as update rules for the MPBP from (7).

IV. Experimental Setup for Performance Evolution of CFNN decoder

For this experiment, a computer system with an $Intel^{©}$ i5 processor with 8GB RAM and a $Windows-10^{©}$ operating system. For simulation purposes, $MATLAB-R-2021b^{©}$ is used.

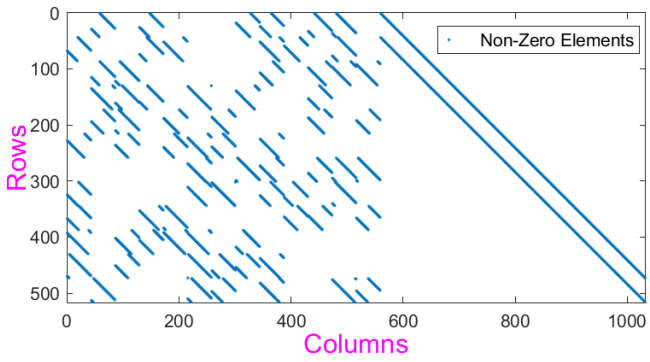

Fig. 5. PC matrix for QC-LDPC

The console generates the PCM based on the input number of rows and code rate. Fig. 5 show the PCM for the QC-LDPC. A random binary stream of 1 s and 0 s is generated to analyze the performance of the CFNN decoder. Using the PCM shown in Fig. 5, QC-LDPC code word were constructed for subsequent transmission via BPSK modulation over the AWGN channel. After reception, the LLR were calculated. The message is then decoded using MPBP decoder, its variants, and CFNN decoder. Finally, the BER was calculated for all decoders across 10,000 blocks within the SNR range of 0 - 5 dB.

V. Results and Discussion over performance of CFNN Decoder

The SNR is pivotal in LDPC code performance analysis. A high SNR prompts the use of high-rate LDPC codes to enhance the transmission efficiency. Conversely, low-rate LDPC codes are preferred under low-SNR conditions to ensure reliable transmissions. Accurate SNR estimation is crucial for achieving high-speed, low-power decoding. Estimating the SNR aids in balancing the performance and complexity when decoding large blocks, which demands significant computation and memory. At low SNR levels, the error probability decreases rapidly, resembling a waterfall curve. However, as the SNR increases, the decrease slows, eventually reaching an error floor at a very high SNR. Understanding the SNR enables system engineers to select appropriate LDPC codes and modulations based on the power and bandwidth constraints. Different modulations affect LDPC code performance differently, making SNR estimation essential for efficient decoding.

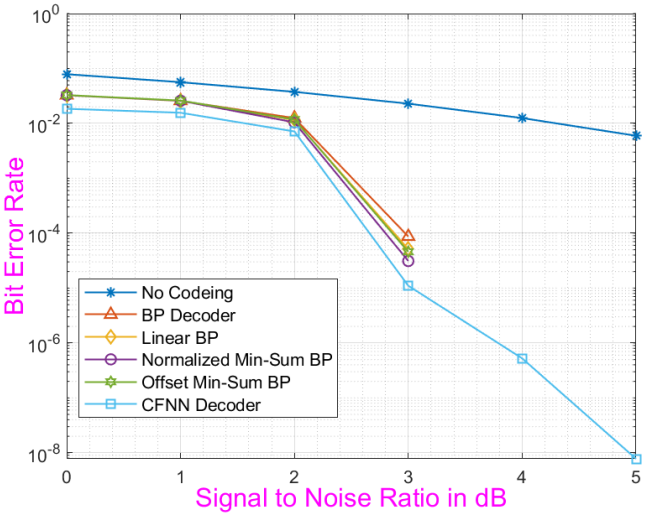

Fig. 6. BER Vs SNR plot with code rate $\frac{1}{2}$, for QC-LDPC code

The illustrations in Fig. 6 provide a comprehensive evaluation of various LDPC decoders, emphasizing their BER performance across different SNR. The absence of coding results in the highest BER at all SNR levels, serving as a baseline for error rates without error-correction coding. Normalized BP demonstrated an improvement over no coding but still exhibited a higher BER compared to other decoding techniques, rendering it less suitable for high-reliability applications. Linear BP performed marginally better than Normalized BP but still lagged behind more advanced decoders. Offset Min-Sum exhibited a significantly lower BER at higher SNR compared to Normalized and Linear BP, demonstrated increased efficacy in error reduction and suitability for scenarios requiring enhanced reliability. The CFNN decoder exhibited the lowest BER among all the methods, particularly at higher SNR, rendering it the most efficient for error

correction and optimal for high-performance communication systems. This analysis shows that advanced decoding techniques, such as the CFNN decoder and Offset Min-Sum BP, significantly outperform conventional methods such as Normalized BP and Linear BP. In particular, the CFNN decoder demonstrated superior performance, suggesting that neural-network-based approaches can substantially enhance LDPC code error correction. These findings are crucial for designing robust communication systems, underscoring the importance of selecting an appropriate decoding algorithm to minimize errors and ensure data integrity. For applications requiring high reliability and performance, the CFNN decoder and Offset Min-Sum BP are recommended because of their superior BER performances at higher SNR.

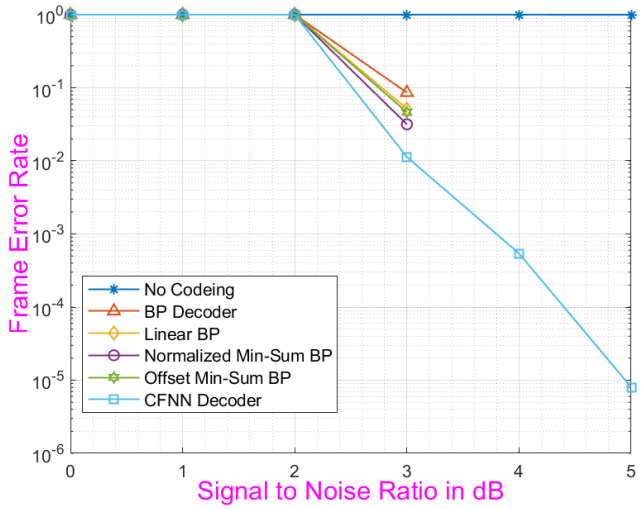

Fig. 7. FER Vs SNR plot with code rate $\frac{1}{2}$, for QC-LDPC code

As Illustrated in Fig. 7 presents a comparative analysis of various LDPC decoders by illustrating the Frame Error Rate (FER) as a function of the SNR in the decibels. The uncoded transmission exhibited the highest FER across all SNR levels, serving as a baseline for comparison. The BP decoder demonstrates improved performance over uncoded transmission; however, it still exhibits a higher FER than the other methods, rendering it less suitable for applications requiring high reliability. The Linear BP marginally outperformed the BP decoder, indicating enhanced error-correction capabilities. The normalized min-sum BP demonstrates a more effective FER reduction at higher SNR compared to BP and Linear BP, making it more appropriate for scenarios demanding increased reliability. The offset min-sum BP achieves a further reduction in the FER at elevated SNR, rendering it suitable for applications requiring enhanced reliability. The CFNN decoder achieved the lowest FER, particularly at higher SNR values, demonstrating superior efficiency in error correction for high-performance communication systems. This analysis suggests that advanced methods, such as the CFNN decoder and offset min-sum BP, significantly outperform traditional methods such as BP and Linear BP. The superior performance of the CFNN decoder indicates that neural-network-based approaches can significantly enhance error correction for LDPC codes.

Table II compares the BER performance of the various decoding methods at an SNR of 3 dB. This analysis highlights the effectiveness of the proposed methods in lowering the BER, which is crucial for improving the accuracy and reliability of the communication systems. The CFNN decoder achieved a BER of 10^{-5} at an SNR of 3 dB, outperforming the other methods. In contrast, the λ-min decoding method explored by Boutillon [40] showed a BER of 10^{-3}, which is two orders of magnitude higher than that of the CFNN decoder. The Information Bottleneck Decoder, as mentioned in Maximilian [41], attained a BER of $10^{-3.8}$ at 3 dB. This method exhibits improved performance compared with decoders, as mentioned earlier. Dinesh [42] experimented that the ANN Decoder exhibits a BER of $10^{-2.1}$ at an SNR of 3 dB, which is higher than that of other techniques. For example, Cheng [45] found that the RC-LDPC Neural Decoding method achieved a BER of $10^{-3.8}$ at the same SNR, similar to the performance of the information bottleneck decoder. The neural layered decoding method proposed by Shah in [46] achieved a BER of $10^{-3.0}$ at an SNR of 3 dB. This indicates that its performance is less effective at higher SNR levels compared to the performance of the CFNN decoder at 3 dB. The analysis shows that the CFNN decoder achieves the lowest BER, suggesting its capability to improve communication system performance relative to existing decoding methods.

TABLE II
COMPARISON OF EXITING METHODS ANALYSIS

Methods	**BER**
Proposed CFNN Decoder	10^{-5} @ 3dB
λMin Decoding [40]	10^{-3} @ 3dB
Information Bottleneck Decoder [41]	$10^{-3.8}$ @ 3dB
Resnet-BP-NN Decoder [44]	$10^{-2.7}$ @ 3dB
ANN Decoder [42]	$10^{-2.1}$ @ 3dB
RC-LDPC Neural Decoding [45]	$10^{-3.8}$ @ 3dB
Neural Layered Decoding [46]	10^{-3} @ 3dB

The training window of the CFNN is shown in Fig. 8. The data set was randomly partitioned into training, validation, and testing subsets utilizing the *dividerand* function. The network employed the conjugate gradient with Powell/Beale restart (*traincgp*) as its training algorithm, while MSE served as the performance metric. Key training state parameters were recorded:

1) Epoch = 376
2) Performance = 0.041019
3) Gradient = 53.7
4) Mu = 0.0001
5) Validation Checks = 6

The network processes the data set 376 times. The current error rate, represented by the MSE performance value, was 0.041019. A gradient of 53.7 indicated substantial ongoing learning. The mu value, a Levenberg-Marquardt algorithm parameter governing the weight adjustments, was 0.0001. Six validation checks were conducted to monitor potential overfitting. The training window presented various analytical plots, including the Performance, Training State, Error Histogram, and Regression, providing comprehensive insights into the network's training progress and performance. The displayed metrics in the training window indicated active learning and improvement in the CFNN training process.

The training performance of the CFNN is shown in Fig. 9, with the number of epochs represented on the horizontal axis

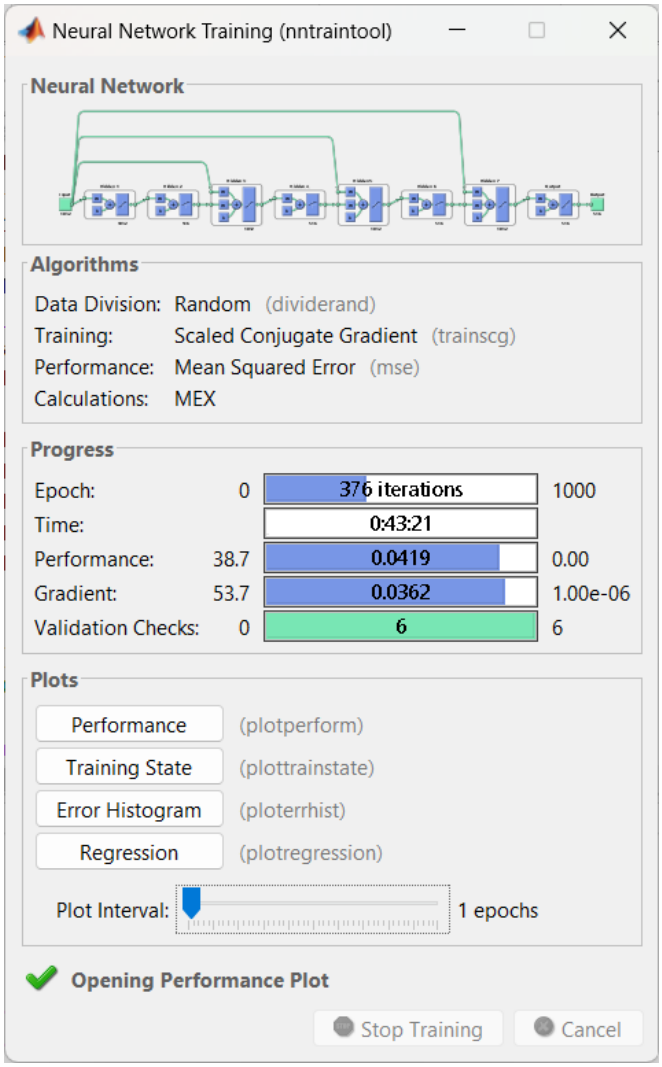

Fig. 8. Training Window showing the training performance of CFNN

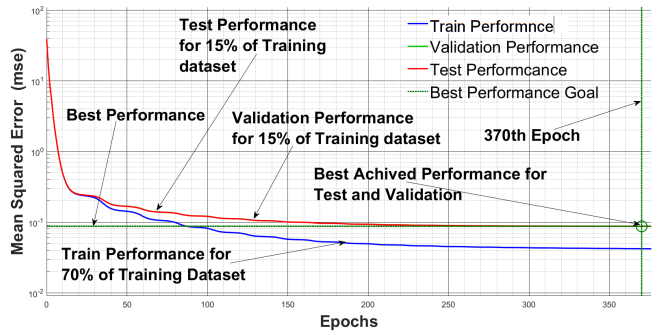

Fig. 9. Training performance of CFNN for Train, Test, and Validation data sets

and the logarithmic scale of Mean Squared Error (MSE) on the vertical axis. The training, test, and validation data sets are denoted by blue, red, and green lines, respectively. The most favorable validation performance, attained at epoch 370 with a MSE of 0.0420 for training and 0.087 for testing and validation, is demarcated by a green circle. The graph exhibits a reduction in MSE over time, signifying the model's learning progression. The simultaneous decline in both training and validation losses suggests the model's capacity to generalize effectively without overfitting. The green line represents the optimal validation performance, while the black dashed line delineates the desired objective. The proximity of the green circle to this line indicates the model's close approximation to the target performance. This visualization provides a comprehensive overview of the training process, highlighting the significant milestone of optimal validation performance at epoch 376.

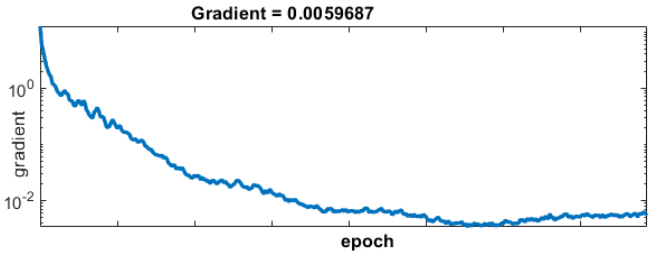

Fig. 10. Gradient variation while training of CFNN

The gradient graph of a CFNN during training is depicted in Fig. 10. The x-axis represents the number of epochs, while the y-axis shows the error gradient on a logarithmic scale. The gradient initiates at a high value and rapidly decreases as the number of epochs increases. At epoch 376, the gradient attains approximately 0.0059687. This substantial decrease over time indicates effective learning and parameter adjustment. The steep initial decline signifies significant early weight updates. As training progresses, the diminishing gradient suggests the model is approaching convergence. The flattening of the curve in later epochs implies minimal changes in gradient from additional weight adjustments. The smooth decline in gradient, without notable fluctuations, indicates stable learning and the absence of instability or learning-rate issues. The decreasing gradient demonstrates effective learning and convergence towards a minimum, while consistent gradient values suggest a well-calibrated and smoothly progressing training process.

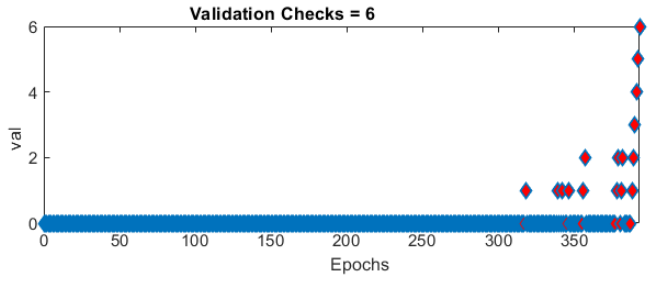

Fig. 11. Validation while training of CFNN

The validation check graph of the CFNN during training is presented in Fig. 11. The horizontal axis depicts epochs ranging from approximately 0 to 400, while the vertical axis represents successful validations from 0 to 6. The blue dots were predominantly concentrated in the lower portion of the graph. At approximately epoch 376, red diamond shapes emerged, indicating an increase in successful validation. Throughout the majority of the training process, the validation checks remained low, with a notable increase occurring around epoch 376, where six validations were recorded. This spike at epoch 376 suggests that the model may exhibit overfitting to the training data, demonstrating

high performance on the training data but poor performance on the validation data, thus potentially compromising generalization. The increased validation checks indicate the necessity for training adjustments such as early stopping, regularization techniques, or learning rate modifications to mitigate overfitting. The successful validations at epoch 376 signify potential overfitting, suggesting suboptimal performance on the unseen data. Modifications to the training process may be required to enhance the generalization capability of the model.

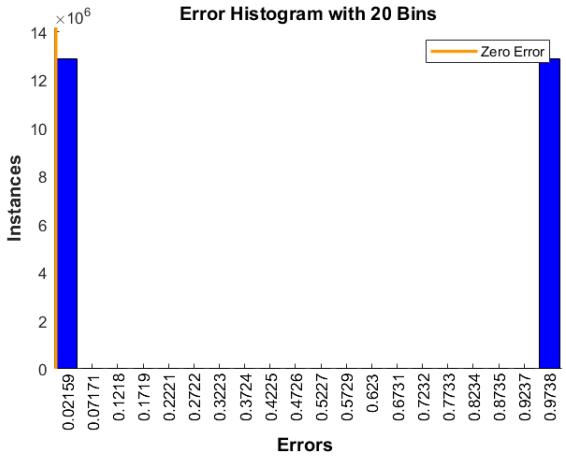

Fig. 12. Validation while training of CFNN

The Fig. 12 presents an error-distribution histogram for a CFNN during the training phase. The horizontal axis represents the error ranges divided into 20 bins, whereas the vertical axis indicates the frequency of instances per bin. Two prominent peaks were observed: one near zero error on the leftmost extremity and another at approximately 0.0738 error on the rightmost extremity. A ”Zero Error” label is positioned adjacent to the leftmost bin. The leftmost peak suggests numerous instances with errors approaching zero, indicating high prediction accuracy for a substantial number of cases. The rightmost peak implies several instances with errors of approximately 0.0738, potentially highlighting outliers or areas of reduced model accuracy. The high frequency of near-zero error instances reflects robust overall model performance. However, the presence of higher errors indicates specific cases with less precise predictions, which necessitates further investigation and model refinement. Strategies such as outlier identification, data augmentation, and hyperparameter optimization, can address these errors. The examination of high-error instances may reveal patterns or features that present challenges for the model. In conclusion, the prevalence of near-zero errors demonstrated satisfactory overall performance, whereas higher errors identified areas for potential improvement.

In Fig. 13 presents four regression plots illustrating the receiver operating characteristics (ROC) of a CFNN during its training phase. These plots encompass the training, testing, validation, and overall stages, with the x -axis representing the false-positive rate and the y -axis showing the true-positive rate. The curves in all four plots exhibit remarkable similarity, forming a linear trajectory from the lower left corner to the upper right corner. This diagonal pattern

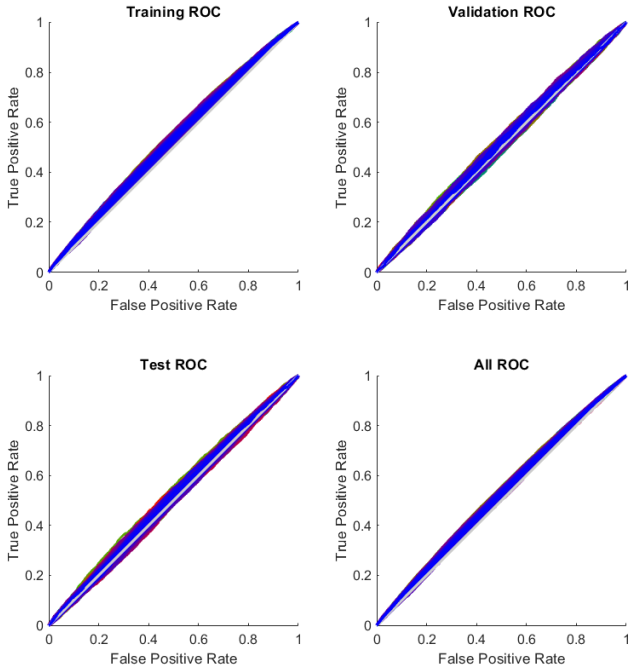

Fig. 13. Validation while training of CFNN

suggests that the capacity of model to identify positives is directly proportional to its false-positive rate across all data sets, performing at a level equivalent to random chance. An optimal ROC curve would demonstrate a convex arc towards the upper left corner, indicating a higher true positive rate coupled with a low false-positive rate. The consistency of the ROC curves across all data splits implies uniform performance; however, the diagonal nature of these lines reveals the model’s inability to effectively discriminate between positive and negative classes. This observation indicates potential issues with the data or model structure, such as insufficient training data, suboptimal feature selection, or inappropriate model architecture. These graphs underscore the failure of the model to differentiate between classes, necessitating a thorough examination of the data preprocessing methods, feature selection techniques, and model architecture to enhance its performance.

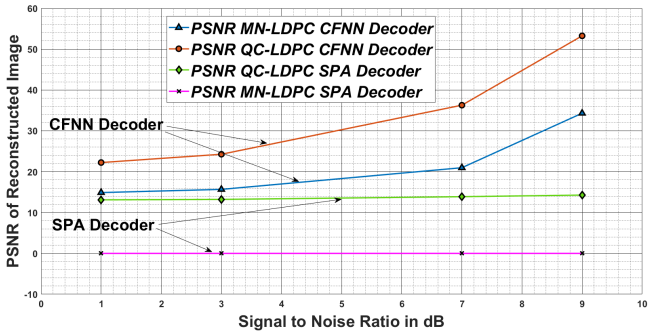

Fig. 14. PSNR plot for Reconstructed Image at SNR = [1,3,7,9]dB

The Fig. 14 depicts PSNR against SNR (dB) for various coding and modulation schemes. As SNR increases from 0 to 10 dB, all lines exhibit an upward trend, indicating PSNR enhancement with higher SNR, as expected due to improved signal quality. PSNR values for QC-LDPC SPA

and QC-LDPC CFNN start near 10 dB at 0 dB SNR, reaching approximately 38 dB at 10 dB SNR, suggesting comparable performance. Conversely, PSNR MN-LDPC CFNN starts notably lower and consistently remains below others, ending near 22 dB at 10 dB SNR, implying inferior performance. Based solely on this graph, QC-LDPC schemes outperform MN-LDPC schemes in terms of PSNR within the given SNR range. The graph facilitates a clear comparison of different schemes, indicating that QC-LDPC schemes generally outperform MN-LDPC ones. The lower performance of MN-LDPC schemes may signify their limitations, especially at lower SNRs, prompting further research. The superior performance of the neural network-based scheme (QC-LDPC CFNN) underscores the potential of machine learning techniques in signal processing tasks, encouraging further exploration in this field.

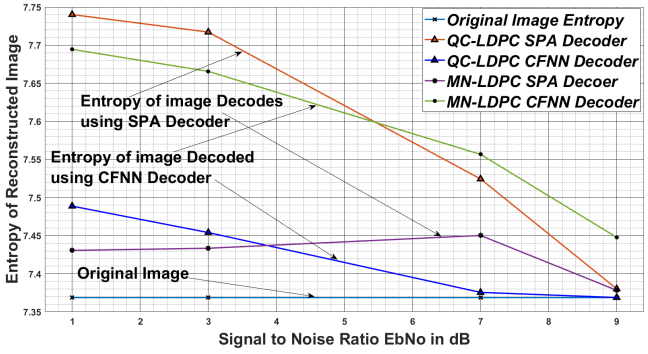

Fig. 15. SNR Vs Entropy Plot Reconstructed Image at SNR = [1,3,7,9]dB

The Fig. 15 illustrates entropy values plotted against SNR in decibels (dB) for various entropy calculations applied to both original and reconstructed images. The entropy of the original image remains relatively constant across different SNR levels, suggesting that its inherent complexity or randomness does not vary with noise levels. Conversely, for the reconstructed images, entropy generally decreases as SNR increases, indicating that higher signal quality corresponds to lower complexity or randomness. This implies that as SNR improves, reconstructed images more closely resemble the original image. Specifically, the QC-LDPC MSPA and QC-LDPC CFNN schemes exhibit similar performance, with their entropy values closely aligned across the SNR range, suggesting comparable entropy reduction. Similarly, the MN-LDPC MP-SPA and MN-LDPC CFNN schemes perform similarly but consistently yield higher entropy values compared to QC-LDPC schemes. This suggests potential differences in effectiveness at entropy reduction, indicating that these schemes may be less adept at reconstructing the original image from noisy data. The insights from this plot shed light on the performance variation among different image reconstruction schemes. It underscores the significant influence of scheme selection on reconstructed image entropy and consequently, on reconstruction quality. However, it's crucial to acknowledge that entropy serves as just one metric of image quality, and other factors may also hold importance depending on the specific application.

VI. Conclusion

This study demonstrates the significant potential of a CFNN as a non-iterative decoder for QC-LDPC code. The CFNN decoder not only addresses the limitations of traditional decoders, such as computational complexity, error floors, and latency, but also achieves superior BER performance. With a notable improvement in BER at various SNR levels, the CFNN decoder proves to be a highly effective alternative, enhancing decoding accuracy and reliability in communication systems. The successful implementation of a CFNN as a non-iterative decoder for QC-LDPC code demonstrates the potential of neural-network-based approaches to address the limitations inherent in traditional decoders. The enhanced BER performance achieved by the CFNN decoder, particularly at lower SNR, indicates that this approach can significantly improve the reliability and efficiency of data transmission across various communication systems. The non-iterative nature of the CFNN decoder may result in reduced computational complexity and latency in the decoding processes, potentially facilitating more rapid and efficient communication in real-time applications. The superior performance of the CFNN decoder in terms of BER and other qualitative and statistical measures suggests that this approach could be particularly advantageous in applications requiring high accuracy and reliability, such as satellite communication or data storage systems. The success of this neural-network-based decoder establishes new research directions for applying machine learning techniques to error correction coding, potentially catalyzing further advancements in coding theory and practice. The improved decoding performance across various SNR levels suggests that CFNN decoders can extend the effective range and capacity of communication systems, potentially enabling a more robust communication in challenging environments or over extended distances.

Acknowledgment

The authors extend their deepest gratitude to the CDRC Lab at VJTI for providing the opportunity to conduct this study. They would like to express our sincere appreciation to the esteemed members of the CDRC Lab for their invaluable support, guidance, and collaboration throughout this endeavor. Their expertise and dedication were instrumental to the success of this research project.

Author Contribution

Harshawardhan Ahire conceived the research idea, conducted simulations, and authored the manuscript. Sushama Wagh supervised the study, verified the results, and edited the manuscript for publication.

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Tran. on Information Theory*, vol. 8, pp. 21–28, 1962.
[2] C. E. Shannon, W. Weaver, and R. E. Blahut, "The mathematical theory of communication," *Urbana: University of Illinois press*, vol. 117, pp. 379–423, 1949.
[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Tran. on Information Theory*, vol. 45, pp. 399–431, 1999.
[4] R. Tanner, "A recursive approach to low complexity codes," *IEEE Tran. on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.
[5] J. Li, S. Lin, K. Abdel-Ghaffar, W. E. Ryan, and D. J. Costello, Jr, *Introduction to LDPC code designs, constructions, and unification*. Cambridge University Press, 2016.
[6] S. Saurabh and K. Sinha, "Ldpc codes based on -resolvable designs," *SN Computer Science*, vol. 4, no. 5, pp. 2661–8907, 2023.

[7] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu, "Construction of non-binary quasi-cyclic ldpc codes by arrays and array dispersions," *IEEE Tran. on Communications*, vol. 57, no. 6, pp. 1652–1662, 2009.

[8] H. Khodaiemehr and D. Kiani, "Construction and encoding of qc-ldpc codes using group rings," *IEEE Tran. on Information Theory*, vol. 63, no. 4, pp. 2039–2060, 2017.

[9] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes: construction based on finite geometries," in *Globecom '00 - IEEE. Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, vol. 2, 2000, pp. 825–829 vol.2.

[10] M. V. Patil, S. Pawar, and Z. Saquib, "Coding techniques for 5g networks: A review," in *2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)*, 2020, pp. 208–213.

[11] M. Jiang, Y. Wang, Y. Lyu, Y. Zhang, X. Xu, and N. Hu, "Flca: A flexible coding approach through 5g nr ldpc codes," *IEEE Communications Letters*, vol. 28, no. 1, pp. 9–13, 2024.

[12] H. Ro and H. Park, "Protograph-based raptor-like ldpc codes with edge addition for urllc," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 2930–2935.

[13] A. Golmohammadi and D. G. M. Mitchell, "Concatenated spatially coupled ldpc codes with sliding window decoding for joint source-channel coding," *IEEE Tran. on Communications*, vol. 70, no. 2, pp. 851–864, 2022.

[14] J. Kim, W. Hur, A. Ramamoorthy, and S. W. McLaughlin, "Design of rate-compatible irregular ldpc codes for incremental redundancy hybrid arq systems," in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1139–1143.

[15] C. Cole, S. Wilson, E. Hall, and T. Giallorenzi, "A general method for finding low error rates of ldpc codes," *CoRR*, vol. abs/cs/0605051, 2006.

[16] L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolic, and M. Wainwright, "Predicting error floors of structured ldpc codes: deterministic bounds and estimates," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 908–917, 2009.

[17] F. Vatta, A. Soranzo, and F. Babich, "More accurate analysis of sum-product decoding of ldpc codes using a gaussian approximation," *IEEE Communications Letters*, vol. 23, no. 2, pp. 230–233, 2019.

[18] S.-Y. Chung, R. Urbanke, and T. Richardson, "Gaussian approximation for sum-product decoding of low-density parity-check codes," in *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, 2000, pp. 318–.

[19] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular ldpc code ensembles," *IEEE Tran. on Information Theory*, vol. 53, no. 1, pp. 39–55, 2007.

[20] J. Chen, R. M. Tanner, J. Zhang, and M. P. C. Fossorier, "Construction of irregular ldpc codes by quasi-cyclic extension," *IEEE Tran. on Information Theory*, vol. 53, no. 4, pp. 1479–1483, 2007.

[21] S. H. Lee and K. S. Kim, "Design of low-rate irregular ldpc codes using trellis search," *IEEE Tran. on Communications*, vol. 57, no. 7, pp. 1994–2004, 2009.

[22] B. Smith, M. Ardakani, W. Yu, and F. R. Kschischang, "Design of irregular ldpc codes with optimized performance-complexity tradeoff," *IEEE Tran. on Communications*, vol. 58, no. 2, pp. 489–499, 2010.

[23] L. Zhang, W. Fu, F. Shi, C. Zhou, and Y. Liu, "A parallel turbo decoder based on recurrent neural network," *Wireless Personal Communications*, vol. 126, no. 2, pp. 975–993, 2022.

[24] C. Condo, M. Martina, and G. Masera, "A network-on-chip-based turbo/ldpc decoder architecture," in *Proceedings of the Conference on Design, Automation and Test in Europe*. San Jose, CA, USA: EDA Consortium, 2012, p. 1525–1530.

[25] L. Chu, H. He, L. Pei, and R. C. Qiu, "Nold: A neural-network optimized low-resolution decoder for ldpc codes," *Journal of Communications and Networks*, vol. 23, no. 3, pp. 159–170, 2021.

[26] L. Guangwen and Y. Xiao, "A recipe of training neural network-based ldpc decoders," 2022.

[27] L. Ma, B. Liu, X. Su, and X. Xu, "A model-driven quasi-resnet belief propagation neural network decoder for ldpc codes," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, 2023, pp. 643–648.

[28] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018.

[29] V. Das, H. Battula, V. Choudhary, R. Ghosh, and M. Priya, "A novel approach on error detection and correction using feedforward artificial neural networks," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Kannur, India, 2019, pp. 639–644.

[30] C.-H. Chang, "Deep and shallow architecture of multilayer neural networks," *IEEE Tran. on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2477–2486, 2015.

[31] S. Han and J. Ha, "A low-complexity neural bp decoder with network pruning," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 1098–1100.

[32] J. Guo, B. Song, Y. Chi, L. Jayasinghe, C. Yuen, Y. L. Guan, X. Du, and M. Guizani, "Deep neural network-aided gaussian message passing detection for ultra-reliable low-latency communications," *Future Generation Computer Systems*, vol. 95, pp. 629–638, 2019.

[33] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[34] R. Tanner, D. Sridhara, A. Sridharan, T. Fuja, and D. Costello, "Ldpc block and convolutional codes based on circulant matrices," *IEEE Tran. on Information Theory*, vol. 50, no. 12, pp. 2966–2984, 2004.

[35] Y. Liang, C.-T. Lam, and B. K. Ng, "A low-complexity neural normalized min-sum ldpc decoding algorithm using tensor-train decomposition," *IEEE Communications Letters*, vol. 26, no. 12, pp. 2914–2918, 2022.

[36] Q. Wang, Q. Liu, S. Wang, L. Chen, H. Fang, L. Chen, Y. Guo, and Z. Wu, "Normalized min-sum neural network for ldpc decoding," *IEEE Tran. on Cognitive Communications and Networking*, vol. 9, no. 1, pp. 70–81, 2023.

[37] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 1361–1365.

[38] N. E. Maammar, S. Bri, and J. Foshi, "Layered offset min-sum decoding for low density parity check codes," in *2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 2018, pp. 1–5.

[39] J. Bellorado and A. Kavcic, "Low-complexity soft-decoding algorithms for reed–solomon codes—part i: An algebraic soft-in hard-out chase decoder," *IEEE Tran. on Information Theory*, vol. 56, no. 3, pp. 945–959, 2010.

[40] E. Boutillon, F. Guillou, and J.-L. Danger, "lambda-min decoding algorithm of regular and irregular ldpc codes," in *3rd International Symposium on Turbo Codes and Related Topics*, 09 2003.

[41] M. Stark, J. Lewandowsky, and G. Bauch, "Information-optimum ldpc decoders with message alignment for irregular codes," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[42] D. Kumar, A. Agrawal, and G. S. Phartiyal, "Non iterative ldpc decoding by syndrome generation using artificial neural network," in *2015 National Conference on Recent Advances in Electronics and Computer Engineering (RAECE)*, 2015, pp. 22–25.

[43] 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.

[44] L. Ma, B. Liu, X. Su, and X. Xu, "A model-driven quasi-resnet belief propagation neural network decoder for ldpc codes," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, 2023, pp. 643–648.

[45] Y. Cheng, W. Chen, L. Li, and B. Ai, "Rate compatible ldpc neural decoding network: A multi-task learning approach," *IEEE Tran. on Vehicular Technology*, vol. 73, no. 5, pp. 7374–7378, 2024.

[46] N. Shah and Y. Vasavada, "Neural layered decoding of 5g ldpc codes," *IEEE Communications Letters*, vol. 25, no. 11, pp. 3590–3593, 2021.

Harshawardhan Ahire (IAENG Member No: 206458) pursued BE in Electronics and Telecommunication Engineering from MGMCOE Nanded, in 2003 and Masters in Electronics Engineering from FrCRCE, Bandra Mumbai, in 2014. He is currently pursuing Ph. D. from Veermata Jijabai Technological Institute, Matunga, Mumbai. He was working with Babasaheb Gawde Institute of Technology, Mumbai From 2002 to 2005. Subsequently, from 2005 to 2008 he was with Rizvi College of Engineering, Mumbai. Afterwords, from 2008 to till date he is working as Assistant Professor at K. J. Somaiya Institute Technology, Sion, Mumbai. while working at K. J. Somaiya Institute of Technology he successfully handled many responsibilities. He has guided more than 100 projects at the undergraduate level. He has approximately 20 years of teaching and research experience, and six years of industry experience. His areas of interest are Error-Correcting codes, Artificial Intelligence, Machine Learning, Computer Vision.

Sushama Wagh (IAENG Member No: 375942) received the B.E. degree from WCE, Shivaji University, the M.E. degree in electrical engineering with a specialization in power systems from Veermata Jijabai Technological Institute (VJTI), Mumbai University, Mumbai, India, and the Ph.D. degree in electrical engineering from the University of Western Australia, Perth, WA, Australia, in 2012. Since 1998, she has been a Faculty Member with VJTI. She is currently a Visiting Scientist with Grid Integration and Mobility Group, SLAC National Accelerator Laboratory, Menlo Park, CA, USA. Before joining SLAC, she was a Visiting Researcher with Tufts University, Medford, MA, USA, from 2015 to 2016, where she was involved in designing dynamic phasor-based controllers for solid-state transformers. Her current research interests include hybrid grid component modeling, analysis, stability, and control.