IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 May 2024, accepted 3 June 2024, date of publication 27 June 2024, date of current version 8 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3420106

RESEARCH ARTICLE

CNN-Based Approach for Enhancing 5G LDPC Code Decoding Performance

SIVARAMA PRASAD TERA[1], RAVIKUMAR CHINTHAGINJALA[2], PRIYA NATHA[3], GIOVANNI PAU[4], (Senior Member, IEEE), C. DHANAMJAYULU[5], (Senior Member, IEEE), AND FARUQ MOHAMMAD[6]

[1]Department of Electronics and Electrical Engineering, Indian Institute of Technology at Guwahati, Guwahati, Assam 781039, India
[2]School of Electronics Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India
[3]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh 522302, India
[4]Faculty of Engineering and Architecture, Kore University of Enna, 94100 Enna, Italy
[5]School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India
[6]Department of Chemistry, College of Science, King Saud University, Riyadh 11451, Saudi Arabia

Corresponding authors: Giovanni Pau (giovanni.pau@unikore.it) and Ravikumar Chinthaginjala (cvrkvit@gmail.com)

The work of Faruq Mohammad was supported by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project under Grant RSP2024R355.

ABSTRACT This paper presents the novel Stochastic decoding-convolutional neural network (SD-CNN) structure, with the goal of enhancing 5G LDPC code's decoding efficiency in the presence of correlated noise of the channel. Applying the stochastic approach acts as an alternate technique to fixed-point LDPC decoding to enhance the decoder's hardware efficiency. In order to improve the efficacy of the decoder, we adopted deep learning method such as Convolutional neural network (CNN). CNNs can be employed for denoising purposes in communication systems, where signals may be compromised by diverse forms of noise while being transmitted, with the aim of enhancing signal quality and dependability. The SD-CNN architecture combines a trained CNN with a stochastic decoder of 5G LDPC codes, thereby utilizing the CNN's capability to accurately estimate channel noise and, consequently, enhance the error correction capabilities of the decoder. The generated output of the trained CNN is then fed back into the stochastic decoder, creating an iterative process between the SD and CNN that ultimately leads to superior decoding performance. For 5G LDPC code word length $N = 3808$, with a base code rate $R = 1/3$, the suggested SD-CNN architecture achieves a BER of 10^{-6} at 0.6 dB of SNR per bit in the strong correlation of channel noise condition, in comparison to SD, which achieves a BER of 10^{-6} at 3.5 dB of SNR per bit. The results demonstrate that there is a 2.9 dB improvement.

INDEX TERMS Channel coding, convolutional neural network (CNN), fifth generation (5G) wireless technology, stochastic decoding (SD), low density parity check codes.

I. INTRODUCTION

Because of their superior error correction capabilities near Shannon's limit [1], [2], LDPC codes [3], [4] [5] have emerged as a crucial channel codes in numerous communication standards. The standards, including 5G New radio (NR) technology [6], IEEE 802.16e (WiMax) [7], 802.11 (WiFi) [8], and DVB-S2 [9]. It is recommended in the 5G New Radio (NR) standards that Quasi-Cyclic (QC) LDPC codes be used for data channels so that they can have high throughput and low latency [10]. To accommodate different coding rates, these QC LDPC codes use two Base Graph Matrices (BGMs) which are BGM-1 (H_{b1}) and BGM-2 (H_{b2}), as well as 51 expansion factors [10]. The H_{b1} can handle code rates between 1/3 and 8/9, and it has 68 block columns and 46 block rows. With a size of 52 block columns and 42 block rows, the H_{b2} is capable of supporting code rates ranging from 1/5 to 2/3.

In the context of 5G standard, a binary LDPC code is defined by the null space of the parity check matrix (PCM) H, which has dimensions of $M \times N$ over GF(2). A bipartite Tanner graph is another graphical representation of the PCM.

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa Zaman Chowdhury.

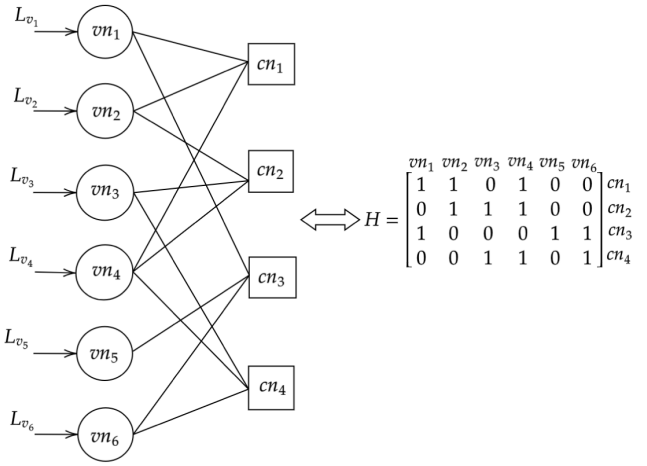


FIGURE 1. A Tanner graph of the LDPC code (6, 2).

Each row of H is made up of M check nodes in the Tanner graph, and each column of H is made up of N variable nodes. The process of decoding LDPC codes is carried out iteratively on its bipartite Tanner graph utilizing the message-passing algorithm such as Sum-product algorithm (SPA) and Min-sum algorithm (MS). Figure 1 displays the PCM H and associated Tanner graph for an example (6, 2) LDPC code. Two different types of nodes make up a Tanner graph. One is the parity or check nodes (CNs), which are represented by the nodes cn_1, cn_2, cn_3, and cn_4. Another is the bit or variable nodes (VNs) are vn_1, vn_2, vn_3, vn_4, vn_5, and vn_6. Depending on the number of ones that are present in the PCM H of the code, these nodes are connected to one another via bidirectional edges. The 5G QC LDPC code's PCM H is built and represented by its BGM H_b. 5G New Radio (NR) specifications demonstrate that the BGM-1 H_{b1} is capable of supporting a variety of coding rates, ranging from 1/3 to 8/9. However, the irregular degrees and diverse connections between the nodes of these BGMs create a more complex inter-node routing network in traditional decoders based on the MS algorithm [12] and SPA [11]. This problem is further aggravated by using multi-bit representation of the Logarithmic-Likelihood Ratio (LLR) symbol messages L_{v_s}, which requires high hardware resources. An alternative approach involves transforming channel probability values P_{v_s} into stochastic bit sequence representations using stochastic approach to decoding [13], [14]. This method simplifies the implementation of logic units and decreases the complexity of inter-node routing [16] in the stochastic decoders. The error correcting capabilities of these stochastic decoders are on par with those of conventional fixed-point decoders [14].

For the purpose of determining the inter-node routing complexity [15], the number of node-interconnects (n_I) of a decoder is computed using (1) expressed as

$$n_I = 2 \times N \times e \times w_v. \tag{1}$$

The length of the codeword is N, the extrinsic message length is e, and the column weight is w_v. As an illustration, SPA and SD based decoders are used to decode a 1/3-rate, N = 3808 LDPC code with an average column weight $w_v = 4.56$. For SPA based decoders, 138,916 node-interconnects of routing are needed, and for SD, 34,729 are needed. The four-fold decrease in node-interconnects in SD can be attributed to extrinsic message length variation in these decoders. While SD uses an extrinsic message length of $e = 1$ due to bit-wise computations in the SD, SPA based LDPC decoders require an $e = 4$.

Stochastic decoding (SD) reduces the inter-node routing complexity compared with Belief propagation(BP) algorithms like MS and SPA. However, using CNN will significantly raise the complexity. Nevertheless, the hardware complexity of SD-CNN is lower than that of the combined SPA and CNN design. The SD-CNN model exclusively employs basic arithmetic operations such as additions, multiplications, and ReLU operations. In contrast, a BP technique incorporates more complex nonlinear operations such as tanh, arctanh, exponential, and logarithms. It would be interesting to do more study in the future on a full complexity comparison based on FPGA and ASIC design.

A. URGENCY OF DEALING WITH COLORED NOISE

In the field of signal processing and communication system related areas, the term ''colored noise'' is used to describe a signal or process that has a power spectrum which is not uniform. This means that the signal has varying levels of energy at different frequencies. Colored noise significantly affects error patterns in transmitted and received signals in communication systems using Low-Density Parity-Check (LDPC) codes. Addressing colored noise is crucial for system performance and reliability.

The satellite communication system faces issues with low-frequency noise due to atmospheric and ionospheric influences. To correct this, adaptive LDPC codes are used. This method dynamically modifies parameters based on real-time channel spectral analysis. For noisy frequencies, a lower code rate is used, while cleaner frequencies require a higher code rate for increased throughput. Wireless sensor networks detect colored noise from machinery and electronic equipment in industrial settings, affecting various electromagnetic spectrum portions. Real-time monitoring adjusts the LDPC coding rate to accommodate fluctuating intensity and spectral properties of industrial noise, ensuring efficient noise detection and management.

5G networks face challenges in addressing colored noise due to high data rates, diverse channel conditions, and advanced modulation techniques. Maintaining accuracy in these conditions is challenging, especially under non-ideal noise conditions. The technology is used in urban and rural areas with dense signal clutter, potentially leading to various forms of colored noise. Accurate noise modeling is crucial for effective decoding in 5G networks.

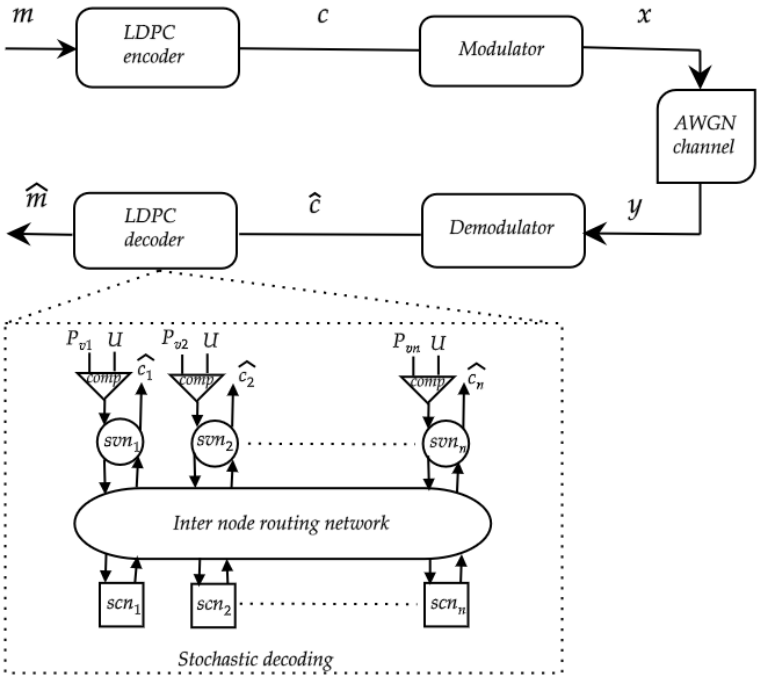


FIGURE 2. A Simplified communication system model.

Noise correlation in communication systems often arises from filtering, oversampling, and device noise. This phenomenon is evident in digital systems, where pink noise is generated by clock jitter [17] and phase noise. LDPC codes face performance degradation when subjected to colored noise, which poses a challenge due to its inherent complexity. One potential solution is the application of whitening, a technique that converts colored noise into white noise. However, this approach necessitates matrix multiplication, which becomes burdensome for lengthy LDPC codes.

Researchers are exploring deep learning techniques on digital communication issues, like channel coding. The development of an SD-CNN architecture to optimize the decoding capability of 5G LDPC codes is inspired from [18]. The architecture combines a trained Convolutional neural network (CNN) with a standard Stochastic decoder, with received symbols processed iteratively between the two. The Stochastic decoder processes the received symbols to acquire the initial decoding results, including the estimated channel noise. This noise passes into a CNN to reduce estimation errors of the SD and get better noise estimation. Iterating between the SD and CNN gradually enhances the SNR per bit E_b/N_o and results in improved decoding.

Our Contributions:

- The key contribution of this work is the proposal of a new Iterative SD-CNN architecture for 5G standard LDPC codes. This design was developed for code word length $N = 3808$, with code rates of $R = 1/3$, 2/5, 1/2, 2/3, 3/4, 5/6, and 8/9 for BGM-1.
- The effects of various loss functions of CNN on our proposed design were also investigated.
- We integrated the field of channel decoders with contemporary deep learning techniques to achieve a 2.9 dB improvement.

The structure of this document is as follows. The foundational ideas of stochastic decoding, colored noise, and the 5G NR LDPC code's Base graph matrix (BGM) are presented in Section II. The suggested design is the main topic of Section III. Results of implementation are given in Section IV. Section V concludes with a note.

II. PRELIMINARIES

A. SIMPLIFIED COMMUNICATION SYSTEM MODEL

The simplified communication system model in the context of channel coding is shown in Fig. 2. In which, the message block vector $\mathbf{m}$, consisting of K bits, is encoded to obtain the codeword $\mathbf{c}$. This codeword is a vector of N bits, achieved by appending $(N - K)$ parity bits to the message block. By utilizing Binary Phase Shift Keying (BPSK) modulation, the modulator generates the modulated symbol vector $\mathbf{x}$. Subsequently, this vector is transmitted over the communication channel and transformed into the received signal vector $\mathbf{y} = \mathbf{x} + \mathbf{n}$ by adding the noise vector $\mathbf{n}$. The channel noise vector $\mathbf{n}$, which has a length of N. The LDPC decoder block is enlarged and shown as a stochastic decoder in Fig. 2. It is made up of comparators (comp) at the stochastic variable nodes (SVN), stochastic check nodes (SCN), and a routing network that connects the nodes.

B. COLORED NOISE

In the context of correlated noise such as colored noise, an Auto-regressive (AR) process of order 1 is used to model noise, where each value in the series is a combination of its previous value and a white noise term. This results in noise with a spectral density that is not flat. In this process, each noise value n_s is defined as

$$n_s = \mathscr{C} \times n_{s-1} + w_s, \tag{2}$$

where n_s is the noise value at time s, $\mathscr{C}$ is the correlation coefficient between consecutive noise samples, with $0 < \mathscr{C} < 1$, and w_s is the white noise at time s with zero mean and variance σ_w^2.

In this process, the variance σ_n^2, assuming the process is stationary variance, and the covariance between any two samples n_i and n_j is given by

$$\sigma_n^2 = \frac{\sigma_w^2}{1-\mathscr{C}^2}, \tag{3}$$

$$Cov(n_i, n_j) = \mathscr{C}^{|i-j|} \times \sigma_n^2. \tag{4}$$

Thus, the entire sequence $\mathbf{n} = [n_1, n_2, \ldots, n_N]^T$ also forms a Gaussian column vector, characterized by its mean and covariance matrix. The covariance matrix is defined by

$$\mathcal{X}_{ij} = \mathscr{C}^{|i-j|} \times \sigma_n^2. \tag{5}$$

The equation (5) describes how the correlation between any two noise values n_i and n_j depends exponentially on the distance $|i-j|$ between their time indices, modulated by the correlation coefficient $\mathscr{C}$.

When the diagonal elements has indices $i = j$, then $\mathcal{X}_{ii} = \sigma_n^2$. This represents the variance of each noise sample n_i, which is σ_n^2, since it includes the entire variance of the white noise added at that step. For off-diagonal elements has indices $i \neq j$, the terms $\mathcal{X}_{ij}$ demonstrate how the correlation decreases as the distance $|i-j|$ increases. Specifically, the correlation coefficient raised to the power of the distance between the indices determines the strength of the correlation. If $\mathscr{C}$ is close to 1, the correlations decay slowly, implying that past values have a substantial influence on future values. Conversely, if $\mathscr{C}$ is close to 0, the correlations decay rapidly, making n_i and n_j almost independent when they are far apart.

The covariance matrix $\mathcal{X}$ is always positive definite if $|\mathscr{C}| < 1$. This ensures that the matrix can be inverted, which is essential for various statistical and signal processing applications, such as computing the likelihoods needed for LLR calculations. $\mathcal{X}$ is symmetric ($\mathcal{X}_{ij} = \mathcal{X}_{ji}$), a property inherent to covariance matrices. This symmetry simplifies mathematical manipulations and computational methods, like Cholesky decomposition, used in simulations and analyses. Since each off-diagonal element depends only on the distance between indices and not on the indices themselves, $\mathcal{X}$ is a Toeplitz matrix. This structure allows for efficient algorithms in matrix computations, which can be exploited in signal processing and machine learning tasks. For the 4-sample scenario, the covariance matrix $\mathcal{X}_4$ is represented as

$$\mathcal{X}_4 = \begin{bmatrix} \sigma_1^2 & \mathscr{C}\sigma_1\sigma_2 & \mathscr{C}^2\sigma_1\sigma_3 & \mathscr{C}^3\sigma_1\sigma_4 \\ \mathscr{C}\sigma_1\sigma_2 & \sigma_2^2 & \mathscr{C}\sigma_2\sigma_3 & \mathscr{C}^2\sigma_2\sigma_4 \\ \mathscr{C}^2\sigma_1\sigma_3 & \mathscr{C}\sigma_2\sigma_3 & \sigma_3^2 & \mathscr{C}\sigma_3\sigma_4 \\ \mathscr{C}^3\sigma_1\sigma_4 & \mathscr{C}^2\sigma_2\sigma_4 & \mathscr{C}\sigma_3\sigma_4 & \sigma_4^2 \end{bmatrix},$$

where σ_1^2, σ_2^2, σ_3^2, and σ_4^2 are the variances of the four noise samples, and $\mathscr{C}$ represents the correlation coefficient between consecutive samples. Assume the case $\mathscr{C} = 0.5$, and the variances of the noise samples $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \sigma_4^2 = \sigma^2 = 1$, and the covariance matrix is simplified as

$$\mathcal{X}_4 = \begin{bmatrix} 1 & 0.5 & 0.25 & 0.125 \\ 0.5 & 1 & 0.5 & 0.25 \\ 0.25 & 0.5 & 1 & 0.5 \\ 0.125 & 0.25 & 0.5 & 1 \end{bmatrix}.$$

For an AR(1) process, the covariance matrix $\mathcal{X}$ is a Toeplitz matrix given by

$$\mathcal{X} = \sigma_n^2 \begin{bmatrix} 1 & \mathscr{C} & \mathscr{C}^2 & \cdots & \mathscr{C}^{N-1} \\ \mathscr{C} & 1 & \mathscr{C} & \cdots & \mathscr{C}^{N-2} \\ \mathscr{C}^2 & \mathscr{C} & 1 & \cdots & \mathscr{C}^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathscr{C}^{N-1} & \mathscr{C}^{N-2} & \mathscr{C}^{N-3} & \cdots & 1 \end{bmatrix},$$

where N is the size of the noise series.

C. LLR CALCULATION IN THE PRESENCE OF COLORED NOISE

The received signal at time s when transmitting BPSK symbols $x_s \in \{+1, -1\}$ is given by

$$y_s = x_s + n_s. \tag{6}$$

The LLR for symbol x_s is calculated by

$$L_{v_s} = \log \frac{P(x_s = +1/y_s)}{P(x_s = -1/y_s)}. \tag{7}$$

Assuming that the noise n_s follows the AR(1) process, and following the Gaussian probability distribution functions as

$$P(y_s/x_s = \pm 1) = \frac{1}{\sqrt{2\pi \det(\mathcal{X})}} \times \exp\left(-\frac{1}{2}(y_s \mp 1)^T \mathcal{X}^{-1}(y_s \mp 1)\right). \tag{8}$$

Computing the LLR as

$$L_{v_s} = -\frac{1}{2}\left[(y_s - \mathbf{1})^T \mathcal{X}^{-1}(y_s - \mathbf{1})\right] + \frac{1}{2}\left[(y_s + \mathbf{1})^T \mathcal{X}^{-1}(y_s + \mathbf{1})\right]. \tag{9}$$

By simplifying this expression as

$$L_{v_s} = -\frac{1}{2}\left[y_s^T \mathcal{X}^{-1} y_s - 2y_s^T \mathcal{X}^{-1}\mathbf{1} + \mathbf{1}^T \mathcal{X}^{-1}\mathbf{1}\right] + \frac{1}{2}\left[y_s^T \mathcal{X}^{-1} y_s - 2y_s^T \mathcal{X}^{-1}\mathbf{1} - \mathbf{1}^T \mathcal{X}^{-1}\mathbf{1}\right]. \tag{10}$$

TABLE 1. An illustration of the generation of stochastic bit sequence.

Cycle of the clock	$u(realvalue)$	U	P	$(P > U)$ output of the comparator
0	0.375	0110	0101	0
1	0.125	0010	0101	1
2	0.25	0100	0101	1
3	0.8125	1101	0101	0
4	0.1875	0011	0101	1
5	0.4375	0111	0101	0
6	0.5625	1001	0101	0
7	0.75	1100	0101	0

Thus, the LLR is

$$L_{v_s} = 2y_s^T \mathcal{X}^{-1}\mathbf{1},$$

where $\mathcal{X}^{-1}$ is the inverse of the covariance matrix of the AR(1) noise process, and $\mathbf{1}$ is a column vector of ones.

D. STOCHASTIC DECODING

In [19], a stochastic method for decoding of LDPC codes was presented. During this decoding procedure, the channel probability values that have been received at each node are transformed into an equivalent stochastic bit sequence. Typically, the LLR of the s-th symbol x_s of $\mathbf{x}$ can be determined by utilizing the $\mathbf{y}$ at the stochastic variable node SVN_s, which is represented as L_{v_s}. The probability of this symbol is calculated as P_{v_s}.

At the initial stage of decoding, the value of P equivalent of P_{v_s} is compared with the Pseudo-Random Number (PRN) U of the Comparator [20] and produces the stochastic bit sequence as shown in Fig. 3. If P is greater than U, then the Comparator will output 1 at that clock cycle; otherwise, it will output 0. For instance, consider the $P_{v_s} = 0.3125$ is denoted as $P = 0101$, which is four-bit fractional symbol. A new four-bit PRN U is generated at each clock cycle by a PRN generator made up of a Linear Feedback Shift Register (LFSR) [21]. The comparator is used to compare $P = 0101$ with newly generated U with every clock cycle. The Comparator produces the stochastic bit sequence 01101000, which corresponds to $P = 0.3125$, as indicated in Table 1. This output is obtained after eight clock cycles. With a mean value of $3/8$, this sequence is in close proximity to $P_{v_s} = 0.3125$ [22]. The generated stochastic bit sequence is bitwise exchanged between the stochastic check nodes (SCN) and the stochastic variable nodes (SVN) throughout the decoding process, continuing until either the desired codeword is discovered or the maximum decoding cycle (DC) limit is reached [23].

In Table 2, an implementation and performance comparison between the SD and alternative LDPC decoders based on normalised min-sum (NMS) and combined min-sum (CMS) is presented. It has been demonstrated that the SD design greatly reduces the complexity of the decoder architecture. With respect to the reported decoders,

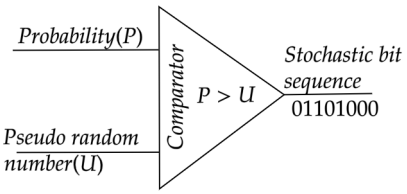


FIGURE 3. Schematic of comparator.

TABLE 2. A comparison of results.

Design	[24]	[25]	[26]	[27]	[28]
The Standard	802.11n	802.15.3c	802.16e	5G-NR	5G-NR
Method of decoding	MS	NMS	NMS	CMS	SD
Code length	1944	672	2304	3808	3808
Shift value	81	21	96	56	56
Code rate	1/2	1/2	1/2	1/3	1/3
Extrinsic Message length (bits)	4	4	6	4	1
DCs or Itrs	10	5	10	10	620
Power (mw)	523	182	870	259	410
Area (mm^2)	4.88	2.25	2.9	1.49	1.10
Throughput (Gbps)	4.5	5.28	2.20	3.04	1.12

the SD design [28] shows an area efficiency gain of 26% over [27], 62% over [26], 51.11% over [25], and 77.36% over [24].

E. THE BASE GRAPH MATRIX (BGM) OF 5G NR LDPC CODE

The BGM H_b [29] is utilized to represent and construct the PCM H of the 5G LDPC code. The dimensions of H and H_b are $M \times N$ and $m_b \times n_b$ respectively. Both the dimensions are related as $N = n_b \times z_c$ and $M = m_b \times z_c$. The lifting size, denoted as z_c, is used to expand the entries of H_b in H using a $z_c \times z_c$ square sub-matrix. There are three types of entry values such as '-1', '0', and non '-1' values are present in H_b. In H, a zero matrix of size $z_c \times z_c$ takes the place of the entry value '-1' from H_b, and the Identity matrix of size $z_c \times z_c$ takes the place of the value '0'. In H, the circulant permutation matrix $I(s_{i,j})$ is used in place of the non '-1' entry value of H_b, which is also called the shift value and ranges from 1 to z_c for all values of $s_{i,j}$. The entry's row and column indices are denoted

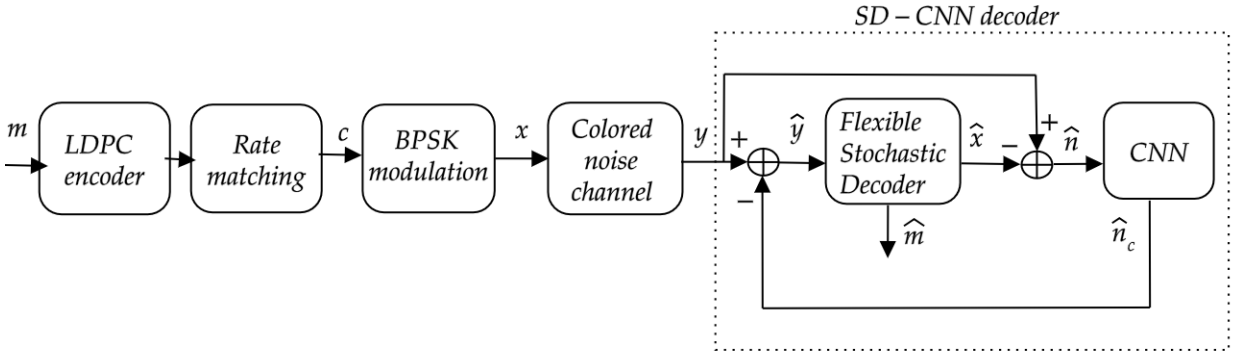


FIGURE 4. Proposed SD-CNN design.

TABLE 3. Parameters and their values of BGM-1.

Parameters of BGM-1	Values
Base code rate	1/3
Edges	316
Lifting size	56
Block columns (n_b)	68
Column weights (w_v)	1 to 30
Block rows (m_b)	46
Row weights (w_c)	3 to 19

by the subscripts i and j, respectively. All rows of the identity matrix are given a right-shift value of $s_{i,j}$ positions to make the sub-matrix $I(s_{i,j})$. Based on the chosen BGM and the set index i_{LS}, the value of $s_{i,j}$ is extracted from Tables 5.3.2-2 and 5.3.2-3 of the 5G-NR standards 3GPP TS 38.212 [10]. The BGM-1 was built in this study using LDPC code with dimension 46×68. The message block length was set at 1232, the code word length was set at 3808, and the code rate was set at 1/3. According to Table 3, the BGM-1 parameters and their values are detailed, and Table 4 displays the constructed BGM-1.

The proposed design incorporates a Rate matching unit to generate the desired coding rate for LDPC codes. Rate Matching Techniques allow the decoders to handle punctured (rate-increased) or shorted (rate-decreased) versions of the mother code without significant loss in decoding performance. This involves algorithms that can reconfigure based on the presence or absence of bits. The suggested SD [28] has runtime flexibility and the capability of decoding received messages corresponding to the set of seven code rates $R = 1/3, 2/5, 1/2, 2/3, 3/4, 5/6$, and $8/9$ and codeword length $N = 3808$ for QC LDPC code compliant to the 5G NR standard. It has been observed that the BER performance improves at lower code rates such as 1/3 and 2/5. Conversely, this performance declines for higher code rates such as 5/6 and 8/9.

III. PROPOSED DESIGN

This section explains the process of the proposed SD-CNN design, as shown in Fig. 4. It also describes the structure of the CNN architecture and the role of the custom cost function in CNN.

A. A FLOW CHART FOR SD-CNN DESIGN

The flow chart of the SD-CNN design process is shown in Fig. 5, which starts at the communication system receiver side. After receiving the signal $\mathbf{y}$, the LLRs and their corresponding probability values of the symbols are calculated as

$$P_{v_s}^{(1)} = P(x_s = +1/y_s) = \left[\frac{e^{L_{v_s}^{(1)}}}{e^{L_{v_s}^{(1)}} + 1}\right]. \tag{11}$$

Similarly, all symbol's calculated values are provided as input to the Stochastic decoder to estimate the symbol vector $\widehat{\mathbf{x}}$. Due to decoding errors, the estimated channel noise $\widehat{\mathbf{n}}$ does not precisely match the actual channel noise $\mathbf{n}$. The estimated channel noise is determined as

$$\widehat{\mathbf{n}} = \mathbf{n} + \mathbf{e}. \tag{12}$$

Here, $\mathbf{e}$ is the error vector of noise. The estimated channel noise $\widehat{\mathbf{n}}$ is fed into trained CNN to generate output as $\widehat{\mathbf{n}_\mathbf{c}}$. The CNN utilizes the correlation property of the channel noise $\mathbf{n}$, which can be regarded as a feature and effectively suppresses the error $\mathbf{e}$ to estimate the accurate channel noise. After obtaining CNN output $\widehat{\mathbf{n}_\mathbf{c}}$ and subtracting it from the received signal $\mathbf{y}$, the new vector $\widehat{\mathbf{y}}$ is calculated as

$$\widehat{\mathbf{y}} = \mathbf{y} - \widehat{\mathbf{n}_\mathbf{c}} = \mathbf{x} + \mathbf{n} - \widehat{\mathbf{n}_\mathbf{c}} = \mathbf{x} + \mathbf{r_c}, \tag{13}$$

where $\mathbf{r_c} = \mathbf{n} - \widehat{\mathbf{n}_\mathbf{c}}$ is residual noise. After that, the updated vector $\widehat{\mathbf{y}}$ is sent through the stochastic decoder again, and another round of decoding is carried out. Prior to the second round of decoding iterations, the following LLR values must be updated as $L_{v_s}^{(2)}$. The superscript (2) of L_{v_s} indicates the LLR's value after it has been processed by CNN. The properties of $\mathbf{r_c}$ will have an impact on the computation of updated LLR values and their consequent impact on

TABLE 4. Base graph matrix.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	...	67
0	55	16	38	35	−1	18	10	−1	−1	0	37	48	21	47	−1	14	14	−1	29	30	48	25	1	0	−1	−1	−1	...	−1
1	29	−1	45	39	46	7	−1	45	21	31	−1	38	37	−1	23	9	6	26	−1	31	−1	19	0	0	0	−1	−1	...	−1
2	39	35	31	−1	8	12	18	39	41	9	14	−1	−1	21	46	21	−1	30	5	55	34	−1	−1	−1	0	0	−1	...	−1
3	33	18	−1	53	5	−1	45	30	16	−1	34	43	45	35	13	−1	40	18	43	−1	30	46	1	−1	−1	0	−1	...	−1
4	2	10	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	...	−1
5	52	3	−1	30	−1	−1	−1	−1	−1	−1	−1	−1	24	−1	−1	−1	14	−1	−1	−1	−1	18	41	−1	−1	−1	−1	...	−1
6	46	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	21	−1	7	−1	−1	−1	51	24	−1	4	−1	−1	−1	−1	−1	−1	...	−1
7	17	20	−1	−1	48	−1	−1	44	38	−1	−1	−1	−1	−1	46	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
8	33	39	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	−1	−1	36	−1	−1	39	−1	2	44	−1	33	−1	−1	...	−1
9	9	37	−1	−1	−1	−1	−1	−1	−1	−1	45	49	−1	33	−1	−1	−1	17	53	−1	50	−1	−1	−1	−1	−1	−1	...	−1
10	−1	26	53	−1	6	−1	−1	19	26	−1	−1	−1	−1	−1	47	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
11	52	11	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	2	−1	−1	−1	35	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
12	30	7	−1	−1	−1	−1	−1	−1	−1	−1	24	3	−1	28	−1	−1	−1	−1	14	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
13	25	−1	−1	0	−1	−1	−1	16	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	−1	22	−1	−1	−1	...	−1
14	14	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	43	23	51	−1	−1	−1	43	−1	−1	−1	−1	−1	...	−1
15	34	8	−1	−1	−1	−1	−1	−1	−1	−1	19	−1	−1	41	−1	−1	−1	−1	41	−1	−1	−1	−1	−1	−1	25	−1	...	−1
16	−1	42	−1	52	−1	−1	−1	−1	−1	−1	−1	43	−1	−1	−1	−1	−1	−1	−1	−1	21	−1	45	−1	−1	−1	−1	...	−1
17	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	54	−1	32	7	−1	−1	−1	4	−1	−1	−1	−1	−1	...	−1
18	−1	31	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	54	32	−1	−1	−1	−1	31	18	−1	−1	−1	−1	−1	−1	...	−1
19	8	6	−1	−1	−1	−1	−1	47	30	−1	8	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
20	49	−1	−1	42	−1	−1	−1	−1	−1	9	−1	46	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	15	−1	−1	−1	−1	...	−1
21	−1	24	−1	−1	−1	19	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	52	−1	−1	−1	50	50	−1	−1	−1	−1	−1	...	−1
22	53	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	3	−1	−1	−1	36	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
23	−1	32	35	−1	−1	−1	−1	−1	−1	−1	0	−1	−1	−1	−1	−1	−1	−1	10	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
24	49	−1	−1	45	8	−1	−1	−1	−1	−1	−1	25	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	12	−1	−1	−1	−1	...	−1
25	−1	1	−1	−1	−1	−1	54	9	−1	−1	−1	−1	−1	−1	25	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
26	51	−1	8	−1	44	−1	−1	−1	−1	15	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
27	−1	40	−1	−1	−1	−1	29	−1	6	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
28	34	−1	−1	−1	41	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	2	−1	−1	−1	−1	−1	...	−1
29	−1	38	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	11	−1	−1	−1	53	−1	−1	2	−1	−1	−1	12	−1	...	−1
30	34	−1	−1	−1	−1	−1	−1	−1	−1	−1	18	−1	−1	42	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
31	−1	7	−1	−1	−1	−1	−1	49	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	9	−1	−1	16	−1	...	−1
32	24	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	41	−1	2	−1	−1	−1	−1	−1	−1	−1	−1	−1	30	−1	−1	...	−1
33	−1	2	49	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	−1	−1	−1	−1	−1	−1	−1	−1	25	−1	−1	−1	−1	−1	...	−1
34	26	−1	−1	−1	−1	−1	18	−1	−1	−1	−1	−1	−1	−1	12	−1	38	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
35	−1	24	−1	−1	−1	−1	5	−1	−1	−1	−1	−1	26	−1	−1	−1	−1	−1	−1	−1	−1	−1	19	−1	−1	−1	−1	...	−1
36	54	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	45	0	−1	−1	6	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
37	−1	25	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	27	−1	−1	−1	−1	−1	−1	−1	−1	−1	26	−1	−1	−1	...	−1
38	11	−1	−1	−1	−1	−1	−1	−1	−1	34	17	−1	10	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
39	−1	12	−1	21	−1	−1	−1	49	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	2	−1	−1	−1	−1	−1	−1	−1	...	−1
40	11	−1	−1	−1	−1	−1	−1	−1	45	−1	−1	−1	−1	−1	−1	−1	−1	40	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
41	−1	23	−1	47	−1	−1	−1	−1	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	55	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
42	2	−1	−1	−1	35	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	22	−1	−1	−1	−1	−1	−1	−1	−1	...	−1
43	−1	38	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	22	−1	22	−1	−1	−1	−1	−1	−1	49	−1	...	−1
44	28	−1	−1	−1	−1	−1	4	−1	9	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	12	−1	−1	−1	−1	...	−1
45	−1	16	−1	−1	−1	−1	9	−1	−1	−1	29	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	...	−1

the efficiency of the following Stochastic decoding process. Hence, it is imperative to train the CNN in a manner that facilitates to accurate estimation of the channel noise by effectively suppresses $\mathbf{r_c}$ by adopting a custom cost function T as shown in (17).

B. STRUCTURE OF CNN FOR NOISE ESTIMATION

The proposed SD-CNN design has utilised a 1-D CNN. It is a variation of the standard CNN designed to process one-dimensional data. The Fig. 6 illustrates the CNN's architecture, which is specified by the number of layers, kernel or filter sizes, and feature maps within each layer. Before beginning the training of the network, certain parameters need to be decided. In most CNN designs, the fully connected, dropout, and max pooling layers are essential. In our architecture, these layers are not taken into account, because the output of the CNN's representation remains at a low level, and its size is the same as that of the CNN input.

The adopted CNN has five convolutional layers. The operation of a 1-D convolutional layer in a network involves the application of kernels to the input data. The input to a 1-D convolutional layer is a one-dimensional sequence of data. After stochastic decoding, the estimated channel noise $\widehat{\mathbf{n}}$ is applied as the input of CNN, a 1-D vector of dimension 3808×1. The convolutional layer contains a set of learnable kernels or filters. Each kernel is a small segment of weights slid across the input data. The filter is convolved, which is element-wise multiplication and summation with the input sequence. This operation is performed by sliding the filter along the input data. The result of the convolution operation is a single value that represents the presence of a particular pattern or feature in the input data. This result is then placed in a new array called the "feature map". In practice, a convolutional layer typically has multiple filters. Each filter can detect a different feature or pattern in the input data. The β-th feature map of α-th layer $f_{\alpha,\beta}$ is calculated as

$$\text{First layer}: f_{1,\beta} = \text{ReLU}(k_{1,\beta} * \hat{n} + b_{1,\beta}), \tag{14}$$

$$\text{Mid layers}: f_{\alpha,\beta} = \text{ReLU}(k_{\alpha,\beta} * f_{\alpha-1} + b_{\alpha,\beta}), \tag{15}$$

$$\text{Last layer}: \widehat{\mathbf{n}_\mathbf{c}} = k_L * f_{L-1} + b_L. \tag{16}$$

The terms $k_{\alpha,\beta}$, $b_{\alpha,\beta}$ are β-th kernel and bias of the α-th layer. Following the convolution process, non-linearity is introduced element-by-element using an activation function such as the Rectified Linear Unit (ReLU) [30]. The output of the convolutional layer is a set of feature maps, one for

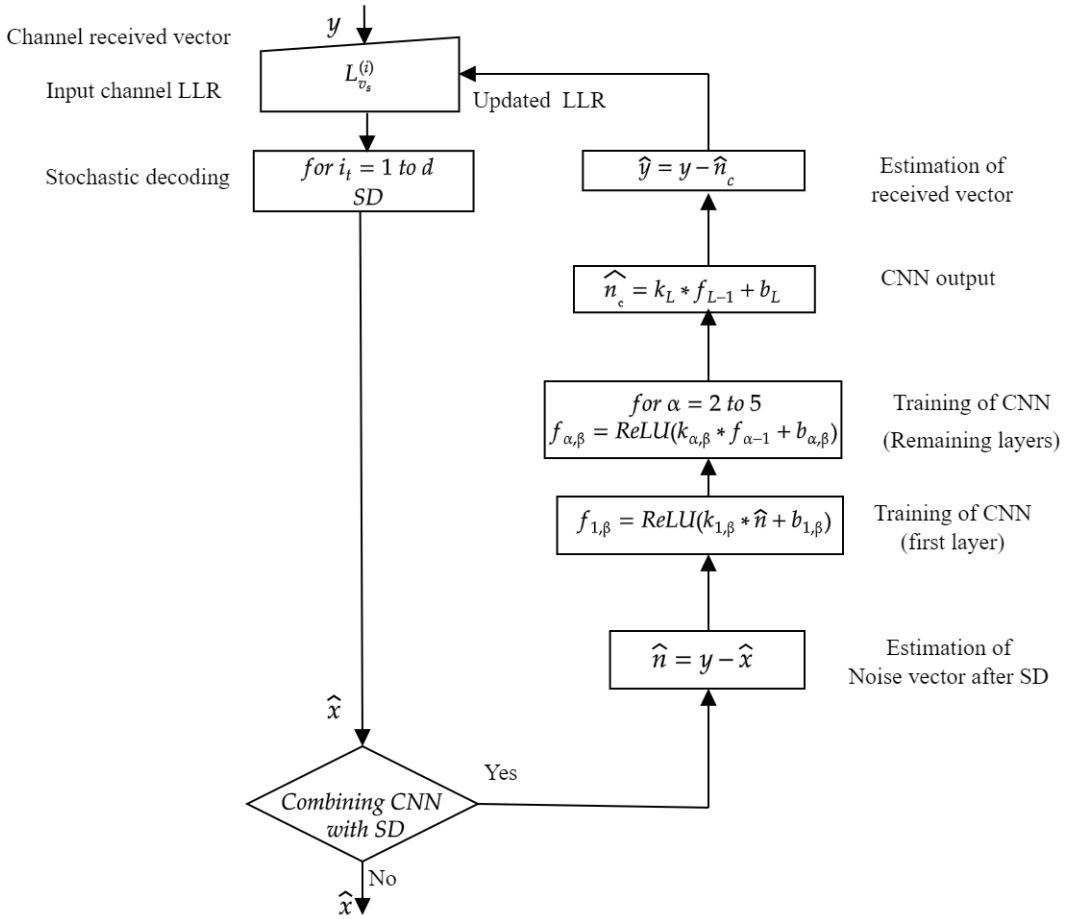


FIGURE 5. SD-CNN flowchart.

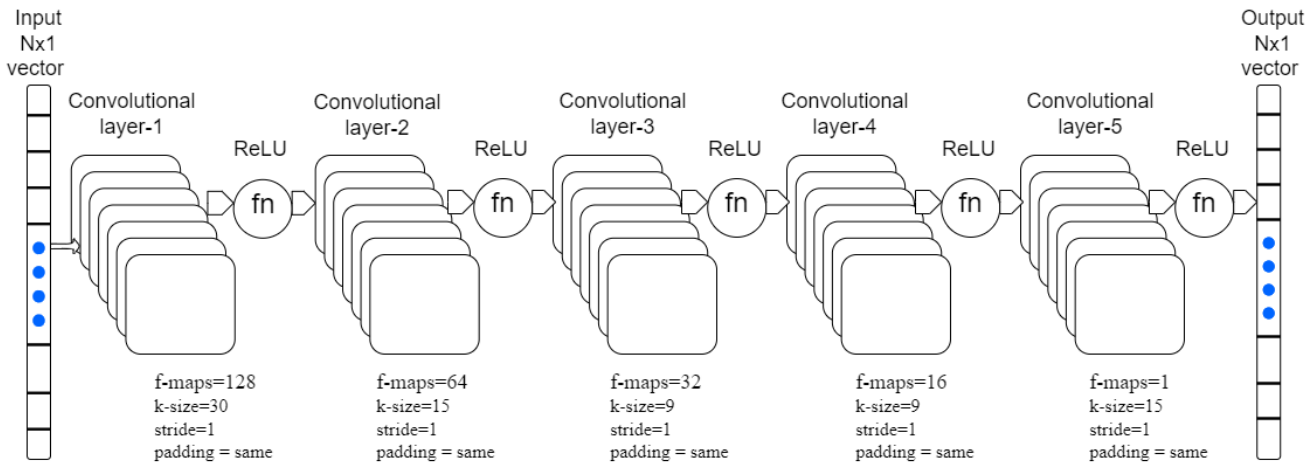


FIGURE 6. Proposed CNN structure.

each filter applied to the input. Each feature map represents the presence of a specific pattern or feature. The process described above is repeated for each filter in the layer, producing a variety of feature maps that each obtain a distinctive aspects of the input data.

1) CUSTOM COST FUNCTION

Custom cost functions refer to functions that the user defined to address specific requirements in a learning task. These functions evaluate the dissimilarity or error between predicted outputs and actual target values during the training

process. In the proposed design, CNN is employed to estimate the channel noise, and its output has an impact on the performance of Stochastic decoding in the subsequent iteration. Consequently, it is crucial to carefully select an appropriate cost function by thoroughly considering the relationship between CNN and the subsequent Stochastic decoding. The newly defined cost function is denoted as

$$T = \frac{||\mathbf{r_c}||^2}{N} + \rho\left(S^2 + \frac{1}{4}(C-3)^2\right), \tag{17}$$

where $\|\mathbf{r_c}\|^2$ is the squared norm of the residual noise, $\mathbf{r_c} = \mathbf{n} - \widehat{\mathbf{n}}_\mathbf{c}$ is residual noise, $\mathbf{n}$ is actual channel noise, $\widehat{\mathbf{n}}_\mathbf{c}$ is the estimated noise by the CNN, N represents the length of the codeword, ρ is a scaling factor that balances the importance of the noise reduction term against the distribution shape terms, and S, C are the skewness and kurtosis of the training data, respectively.

The initial term in equation (17) measures the power of the residual noise, while the second term, borrowed from the Jarque-Bera test [31], functions as a normality test to assess the extent to which a given data set follows a Gaussian distribution. The scaling factor ρ is employed to strike a balance between these two objectives. We adopted a custom cost function that includes terms for skewness (S) and kurtosis (C) of the training data. The cost function aims to minimize not only the residual noise but also adjusts the distribution of estimated noise to closely match a normal distribution by considering its skewness and kurtosis. The Squared Norm of Residual Noise term ($\frac{\|\mathbf{r_c}\|^2}{N}$) aims to minimize the average power of the residual noise across all observations, pushing the network towards accurate noise estimation. The regularization term $\rho\left(S^2 + \frac{1}{4}(C-3)^2\right)$ adds a penalty for deviations from the normal distribution characteristics. The Skewness Squared (S^2) term penalizes asymmetry in the noise estimates. The Excess Kurtosis Squared ($\frac{1}{4}(C-3)^2$) penalizes deviation from the normal level of tailedness. The steps to implement this cost function in the training of a CNN are

- Compute residual noise $\mathbf{r_c}$: Calculate the difference between actual and estimated noise for each sample.
- Calculate Skewness and Kurtosis: Utilize statistical functions to compute the skewness and kurtosis of the estimated noise over the training batch.
- Combine in Cost Function: Use the defined cost function T to compute the loss during each training step.
- Optimize: Use an optimization algorithm (like SGD or Adam) to minimize $\mathbf{r_c}$ across training epochs, adjusting the network weights accordingly.

Skewness measures the asymmetry of a data distribution around its mean. It is typically calculated using the following formula:

$$S = \frac{\frac{1}{N}\sum_{i=1}^{N}(r_i - \overline{r})^3}{\left(\frac{1}{N}\sum_{i=1}^{N}(r_i - \overline{r})^2\right)^{3/2}}, \tag{18}$$

where r_i are the individual observations, and $\overline{r}$ is the mean of the observations. This formula takes the cube of each observation's deviation from the mean (measuring asymmetry), normalizes it by the cubed standard deviation (to make it dimensionless), and averages it across all observations.

Kurtosis measures the tailedness of the distribution, focusing on the tails relative to the rest of the data. The formula for kurtosis used in many statistical applications is:

$$C = \frac{\frac{1}{N}\sum_{i=1}^{N}(r_i - \overline{r})^4}{\left(\frac{1}{N}\sum_{i=1}^{N}(r_i - \overline{r})^2\right)^2}. \tag{19}$$

This formula measures how much of the data is in the tails: a higher kurtosis indicates heavier tails (more outliers), and a lower kurtosis indicates lighter tails.

2) CHANNEL NOISE IN TRAINING

Both the estimated noise $\widehat{\mathbf{n}}$ data from the results of stochastic decoding and the channel noise $\mathbf{n}$ data are required to ensure the network is trained effectively. We focus on the receiver-recognized noise correlation functions of channel models [32]. This allows us to produce adequate channel noise samples for use in training the network. These samples are produced as a result of the below relationship.

$$\mathbf{n} = \mathcal{X}^{\frac{1}{2}}\mathbf{n_g}, \tag{20}$$

where $\mathbf{n_g}$ is a vector of Gaussian random variables with a standard distribution that are independent and identically distributed (i.i.d).

3) CONVOLUTION OPERATION IN CNNS

The convolution operation in a CNN applied to colored noise data involves sliding a filter (kernel) across the input noise vector. For each position of the filter, a dot product is computed between the filter weights and the segment of data it covers. Assume a filter with weights $[g_1, g_2, \ldots, g_k]$ of size k, and a segment of the AR(1) series covered by the filter at time t is $[n_{t-k+1}, \ldots, n_t]$. The convolution at this position is calculated as

$$V_t = g_1 \cdot n_{t-k+1} + g_2 \cdot n_{t-k+2} + \cdots + g_k \cdot n_t. \tag{21}$$

This result, V_t, is the output for that filter position, contributing to one element of the feature map. After computing the convolution, the ReLU (Rectified Linear Unit) activation function is applied to the convolution output. Applying ReLU to V_t yields

$$R_t = \max(0, V_t). \tag{22}$$

The convolution operation combined with the ReLU activation can effectively highlight regions of autocorrelation in the time series. If the series is positively auto-correlated and $\mathscr{C}$ is close to 1, successive values of the series are similar and positive. Suppose the filter weights are designed to detect this similarity (e.g., all positive). In that case, the convolution V_t

TABLE 5. CNN parameters.

Parameter	Value
Number of Layers	5
Filter Size	{30, 15, 9, 9, 15 }
Number of Feature maps	{128, 64, 32, 16, 1}
Padding	Not applicable
Pooling layers	Not applicable
Fully Connected Layers	Not applicable
Activation Function	ReLU
Optimizer	Adam
Loss Function	$\frac{\|\|r\|\|^2}{N} + \rho\left(S^2 + \frac{1}{4}(C-3)^2\right)$
Weight Initialization	He Normal
Training data size	17,50,000
Testing data size	80,000
Accuracy	94.5%
Number of epochs	30

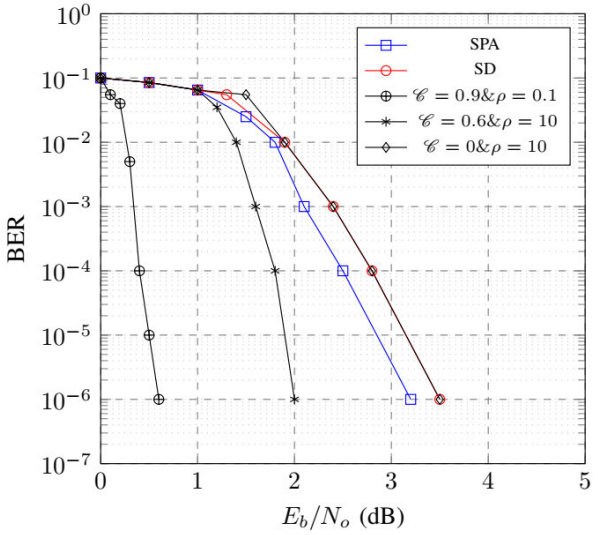


FIGURE 7. BER plot of various algorithms.

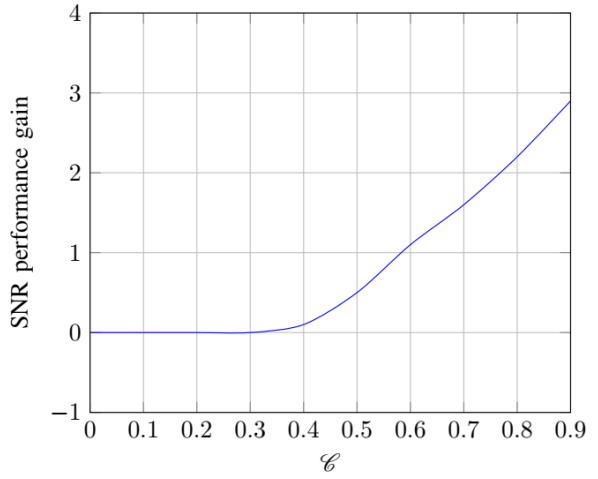


FIGURE 8. Relation between performance gain and correlation coefficient.

at positions where X_t and X_{t-1} are both positive will also be positive and significant, thus surviving the ReLU activation. This means R_t will be high in regions where the series exhibits strong positive autocorrelation, effectively highlighting these regions in the feature map.

IV. IMPLEMENTATION RESULTS

A. APPROACH

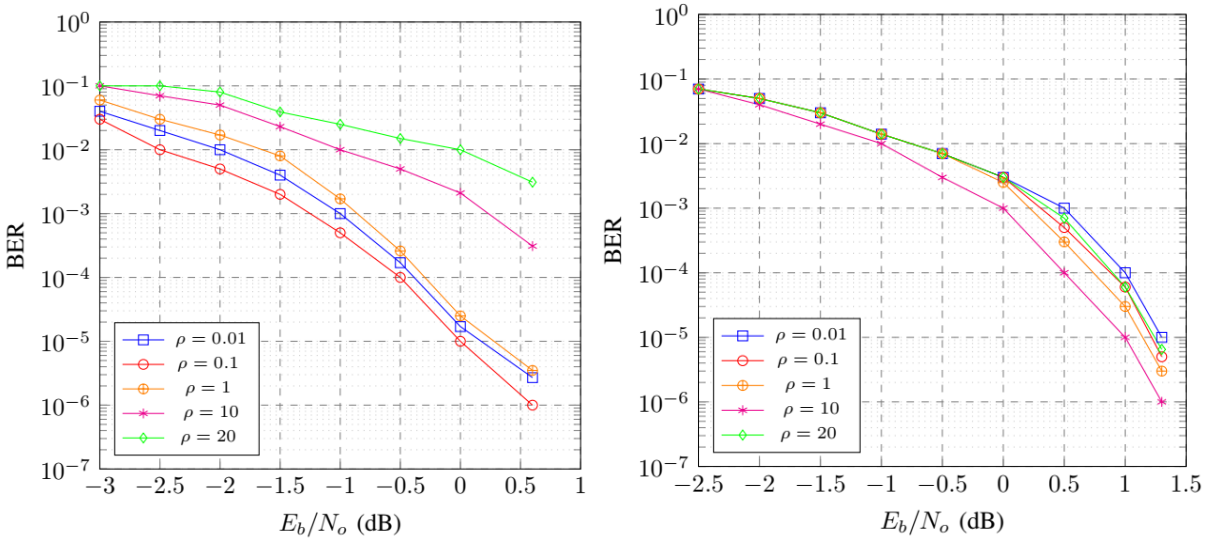
To validate the design, the 5G LDPC code word length $N = 3808$ of BGM1 with base code rate $R = 1/3$ is used. Google Colab, MATLAB, and Tensor Flow [33] are used to build the simulation platform. A neural network's weights can be initialized [34] using a method called the He Normal initialization. This method initializes the weights in such a way as to facilitate the neural network's capacity for efficient learning. Prior to training the network on the training data, it must be generated. To test the network cost function and prevent possible over fitting, some validation data is also generated, as is standard procedure in machine learning. The training data are produced using a variety of signal-to-noise power (SNR)s, including 0 dB, 0.5 dB, 1 dB, 1.5 dB, 2 dB, 2.5 dB, 3 dB, 3.5 dB and 4 dB. Each SNR's data takes up the same percentage of the total data. The network is trained using a classic mini-batch gradient descent approach. There are 1200 blocks of data in each mini-batch, and each SNR's data takes up the same amount of space. To find the gradient, a fixed number of training samples, called a "mini-batch," are picked at random in each iteration. When searching for the optimal parameters for the network, we make use of the Adaptive Moment Estimation (Adam) optimization method [35]. Training is continued in our simulation until the loss does not reduce over an extended length of time. The CNN parameters and their values are described in the Table 5. One measure of a system's performance is its bit error rate (BER). This measurement takes into account the channel's SNR E_b/N_o at a specified BER of 10^{-6}, which is the decoder's transmission energy efficiency. One hundred frame errors per BER measurement is required by the simulations.

B. EVALUATION OF PERFORMANCE

The performance of the design depends on three parameters such as the correlation coefficient $\mathscr{C}$, the scaling factor ρ, and the number of repetitions between SD and CNN.

1) THE CHOICE OF CORRELATION COEFFICIENT

This study focuses on three scenarios in which the correlation coefficient of the channel noise $\mathscr{C}$ was either 0.9 indicating a high correlation, 0.6 indicating a moderate correlation, or 0 indicating no correlation. Simulation results focused on the suggested one-iteration of the SD-CNN decoder (SD-CNN)1. As shown in Fig. 7, when $\mathscr{C} = 0.9$, the decoding performance of SD-CNN improved by about 2.9 dB at a BER of 10^{-6}. The performance gain is smaller when the



(a) All tested ρ values when $\mathscr{C}$ = 0.9. (b) All tested ρ values when $\mathscr{C}$ = 0.6

FIGURE 9. BER plot of (SD-CNN)1 for all tested ρ values.

correlation is moderate $\mathscr{C} = 0.6$. It demonstrates that when the correlation is weaker, using a CNN has less impact. When $\mathscr{C} = 0$, the proposed design works similarly to standard stochastic decoding (SD), in which the noise samples are i.i.d. The results lead us to conclude that the iterative SD-CNN decoding method works well across various of correlation coefficients, with performance gains that vary adaptively depending on the correlation of the noise. The performance gain improves steadily as correlation level $\mathscr{C}$, as expected, because a higher $\mathscr{C}$ allows CNN to extract the correlation of noise as a feature. The performance gain of proposed decoder increases when there is increase in correlation coefficient of the channel noise $\mathscr{C}$ as shown in Fig. 8.

2) THE SELECTION OF SCALING FACTOR

The study presents a custom cost function T that measures normality component and power of residual noise $\mathbf{r_c}$. The value of ρ in T, plays a significant role in achieving a balance between the power of $\mathbf{r_c}$ and its distribution. From simulations, it has been observed the very little values of ρ are unable ensure a Gaussian distribution for the $\mathbf{r_c}$, while extremely high values of ρ are unable to minimise the residual noise power. Therefore, an appropriate selection of ρ affects the performance of SD-CNN.

The results of adopting various values of the scaling factor ρ at two distinct channel correlation coefficients $\mathscr{C} = 0.9$ and 0.6, using an architecture of (SD-CNN)1, are shown in the Fig. 9. The Table 6 displays the results for two different values of $\mathscr{C}$. When $\mathscr{C}$ is 0.9, a Bit Error Rate (BER) of 10^{-6} is reached at $\rho = 0.1$. On the other hand, when $\mathscr{C}$ is 0.6, a BER of 10^{-6} is achieved at $\rho = 10$. The value $\rho = 0.1$ is performing best for high correlation case $\mathscr{C} = 0.9$ and $\rho = 10$ for moderate correlation case $\mathscr{C} = 0.6$. The smaller ρ of T is preferred for high correlation coefficient $\mathscr{C}$ which makes residual noise distribution approximately follow the Gaussian distribution. On the other hand, when correlation weakens and input elements become more independent, it is preferable to have a larger ρ to better address residual noise distribution.

TABLE 6. BER values of (SD-CNN)1 for all tested ρ values.

	$\mathscr{C}$ = 0.9 BER at 0.6dB	$\mathscr{C}$ = 0.6 BER at 1.3dB
ρ=0.01	4×10^{-5}	10^{-5}
ρ= 0.1	10^{-6}	5×10^{-6}
ρ=1	5×10^{-5}	4×10^{-6}
ρ=10	5×10^{-4}	10^{-6}
ρ=20	5×10^{-3}	8×10^{-6}

3) IMPACT OF NUMBER OF REPETITIONS BETWEEN CNN AND SD

Previously, the simulation results focused on the suggested one-iteration of the SD-CNN decoder, which is represented as (SD-CNN)1. In order to further improve the performance gain, we may naturally do many iterations between CNN and SD. At BER = 10^{-6}, the decoding performance may be improved by 0.5 dB using four iterations between CNN and SD, which is represented as (SD-CNN)4 compared to (SD-CNN)1 as shown in Fig. 10. From Table 7, the performance increase also becomes negligible beyond four iterations of the SD-CNN. It is due to the CNN having hit its limit and being unable to reduce the remaining noise power any more.

C. IMPACT OF VARIOUS LOSS FUNCTIONS

Loss functions are broadly divided into two types: regression loss and classification loss. In our design results are regression like predicting the continuous values. Here, regression loss is applied. When assessing the efficiency of ML models, especially for regression tasks, two popular metrics are Mean Absolute Error (MAE) and Mean Squared Error (MSE) [36].

TABLE 7. Estimation of SNR (dB) at BER of 10^{-6} from various repetitions between CNN and SD.

Correlation coefficient($\mathscr{C}$)	Scalar factor(ρ)	For (SD-CNN)1	For (SD-CNN)2	For (SD-CNN)4
0	10	3.5	3.2	3
0.6	10	1.3	1	0.8
0.9	0.1	0.6	0.3	0.1

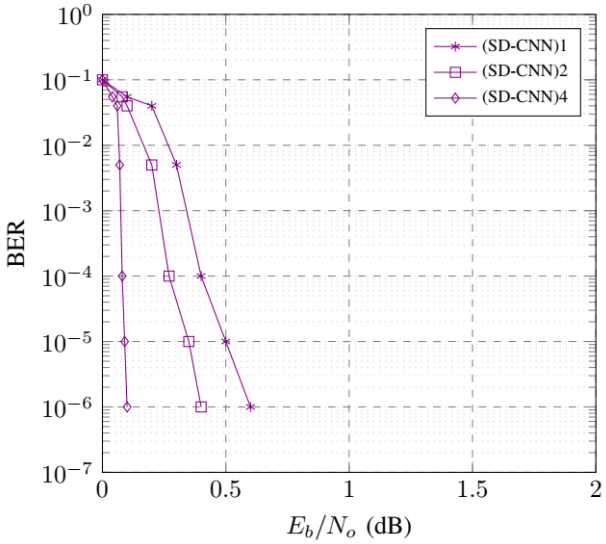


FIGURE 10. BER plot of various iterations between SD-CNN design when $\mathscr{C} = 0.9$, $\rho = 0.1$.

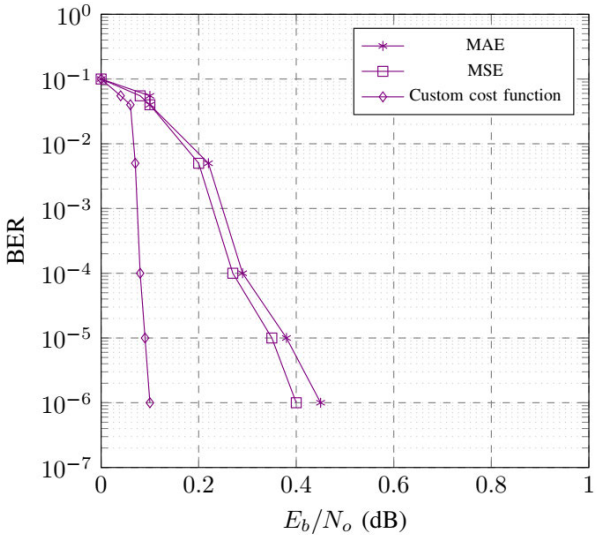


FIGURE 11. BER plot of (SD-CNN)1 design of various loss functions.

These metrics assist in determining how well the predictions match the actual data points by measuring the difference between the predicted and actual values. The Mean Squared Error (MSE) is a measure of how close an estimate is to the true value, or how far off it is. It is calculated by averaging the squares of all of the errors. The formula for MSE is

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(n_i - \hat{n}_\mu)^2, \tag{23}$$

where n_i is the i^{th} element in the noise vector, $\hat{n}_\mu$ is the sample mean, and N is size of the coded block. MSE is sensitive to outliers as squaring the errors amplifies the effects of these points on the error metric. MAE measures the average magnitude of the errors in a set of predictions, without considering their direction (i.e., it takes the absolute value of each error). The formula for MAE is

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|n_i - \hat{n}_\mu|, \tag{24}$$

where $|n_i - \hat{n}_\mu|$ is the absolute error between the actual values and the predicted values, N is size of the coded block. MAE is particularly useful because it provides a direct average error magnitude from the model predictions. We considered three main types of loss functions, such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and the proposed Custom cost function T. The proposed cost function is a technique used to maintain the balance between power and distribution of residual noise. The study involves training of the (SD-CNN)1 model using three different loss functions. The simulated results are then observed in a Fig. 11, using a scaling factor of $\rho = 0.1$, channel correlation coefficient $\mathscr{C} = 0.9$, and one iteration between SD and CNN. From Fig.11, the performance gain improved by 0.3 dB by considering the custom cost function T when compared other loss functions.

V. CONCLUSION

To address channel correlated noise, we have developed a unique SD-CNN decoding architecture in this study. The suggested architecture consists of iterating between a CNN and a stochastic decoder once they have been sequentially concatenated together. The purpose of the stochastic decoder is to make an estimate of the bits that have been coded as well as an indirect estimate of the channel noise. By understanding the noise correlation, the CNN will be able to eliminate the channel noise estimation mistakes of the stochastic decoder. Extensive simulations were performed to demonstrate the efficacy of the proposed SD-CNN decoder.

ABBREVIATIONS AND ACRONYMS

5G NR	Fifth Generation New radio
AWGN	Additive White Gaussian Noise
BER	Bit error rate
BGM	Base Graph Matrix
BPSK	Binary Phase Shift Keying

CMS	Combined min-sum
CNN	Convolutional neural network
E_b/N_o	Signal-to-noise ratio per bit
i.i.d	independent and identically distributed
LDPC	Low density parity check codes
LFSR	Linear Feedback Shift Register
LLR	Logarithmic-Likelihood Ratio
MAE	Mean Absolute Error
MS	Min-sum algorithm
MSE	Mean Squared Error
NMS	Normalised min-sum
PCM	Parity check matrix
PRN	Pseudo-Random Number
QC	Quasi-Cyclic
SCN	Stochastic check nodes
SD	Stochastic decoding
SNR	Signal-to-noise ratio
SPA	Sum-product algorithm
SVN	Stochastic variable nodes

ACKNOWLEDGEMENT

The funding from Researchers Supporting Project number (RSP2024R355), King Saud University, Riyadh, Saudi Arabia.

REFERENCES

[1] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.
[2] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, 2001.
[3] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, 1962.
[4] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[5] N. Wiberg, "Codes and decoding on general graphs," Dept. Elect. Eng. Linköping Univ., Linöping, Sweden, Tech. Rep. , 1996.
[6] *3GPP TSG RAN WG1 Meeting 86Bis*, document R1-1611081, 3GPP, Lisbon, Portugal, Oct. 2016, pp. 83–89.
[7] *Standard for Local and Metropolitan Area Networks—Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, Standard IEEE 802.16-2004, 2004.
[8] *Standard for Information Technology-Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Standard IEEE 802.11n-2009, 2009.
[9] ETSI, *Digital Video Broadcasting (DVB); Second Generation*, Standard ETSI EN 302 307, V1.3.1, 2013.
[10] Ad-Hoc Chair (Nokia). *Chairman's Notes of Agenda Item 7.1.4. Channel Coding*, document TSG RAN WG1 Meeting AH 2, R1-1711982, 3GPP, 2017. [Online]. Available: https://portal.3gpp.org
[11] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, 2001.
[12] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2150–2158, Jul. 2014.
[13] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *Electron. Lett.*, vol. 39, no. 3, pp. 299–301, 2003.
[14] X.-R. Lee, C.-L. Chen, H.-C. Chang, and C.-Y. Lee, "A 7.92 Gb/s 437.2 mW stochastic LDPC decoder chip for IEEE 802.15.3c applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 2, pp. 507–516, Feb. 2015.
[15] V. A. Chandrasetty and S. M. Aziz, *Resource Efficient LDPC Decoders: From Algorithms to Hardware Architectures*. Cambridge, MA, USA: Academic Press, 2017.
[16] J. P. Hayes, "Introduction to stochastic computing and its challenges," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2015, pp. 1–3.
[17] A. Hajimiri and T. H. Lee, "A general theory of phase noise in electrical oscillators," *IEEE J. Solid-State Circuits*, vol. 33, no. 2, pp. 179–194, 1998.
[18] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," in *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 144–159, Feb. 2018.
[19] W. J. Gross, V. C. Gaudet, and A. Milner, "Stochastic implementation of LDPC decoders," in *Proc. 39th Asilomar Conf. Signals, Syst. Comput.*, 2005, pp. 713–717.
[20] D. Zhang and H. Li, "A stochastic-based FPGA controller for an induction motor drive with integrated neural network algorithms," *IEEE Trans. Ind. Electron.*, vol. 55, no. 2, pp. 551–561, Feb. 2008.
[21] A. Dinu, M. N. Cirstea, and M. McCormick, "Stochastic implementation of motor controllers," in *Proc. IEEE Int. Symp.*, vol. 2, Jul. 2002, pp. 639–644.
[22] S. Sharifi Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic LDPC decoders," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5692–5703, Nov. 2008.
[23] C. Winstead, V. C. Gaudet, A. Rapley, and C. Schlegel, "Stochastic iterative decoders," in *Proc. Int. Symp. Inf. Theory*, 2005, pp. 1116–1120.
[24] I. Tsatsaragkos and V. Paliouras, "A reconfigurable LDPC decoder optimized for 802.11 n/ac applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 1, pp. 182–195, Jan. 2018.
[25] M. R. Li, C. H. Yang, and Y. L. Ueng, "A 5.28-Gb/s LDPC decoder with time-domain signal processing for IEEE 802.15.3c applications," *IEEE J. Solid State Circuits*, vol. 52, no. 2, pp. 592–604, Feb. 2017.
[26] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 985–994, Aug. 2009.
[27] T. T. B. Nguyen, T. N. Tan, and H. Lee, "Low-complexity high-throughput QC-LDPC decoder for 5G new radio wireless communication," *Electronics*, vol. 10, no. 4, p. 516, 2021.
[28] S. P. Tera, R. Alantattil, and R. Paily, "A flexible FPGA-based stochastic decoder for 5G LDPC codes," *Electronics*, vol. 12, no. 24, p. 4986, Dec. 2023.
[29] M. P. C. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[31] T. Thadewald and H. Buning, "Jarque-Bera test and its competitors for testing normality—A power comparison," *J. Appl. Statist.*, vol. 34, no. 1, pp. 87–105, 2007.
[32] S. K. Sharma, S. Chatzinotas, and B. Ottersten, "SNR estimation for multi-dimensional cognitive receiver under correlated channel/noise," *IEEE Trans. Wireless Commun.*, vol. 12, no. 12, pp. 6392–6405, Dec. 2013.
[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement.*, 2016, pp. 265–283.
[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[36] F. Chahkoutahi and M. Khashei, "Influence of cost/loss functions on classification rate: A comparative study across diverse classifiers and domains," *Eng. Appl. Artif. Intell.*, vol. 128, Feb. 2024, Art. no. 107415.

SIVARAMA PRASAD TERA received the B.Tech. degree in electronics and communication engineering from Jawaharlal Nehru Technology University, Kakinada, India, and the M.Tech. degree in digital systems from NIT Allahabad, India. He was an Assistant Professor with KL University, Vijayawada, India. He is currently a Research Scholar with the Department of Electronics and Electrical Engineering, IIT Guwahati, India. His research interests include machine learning, deep learning, communication systems, and VLSI devices/circuits.

RAVIKUMAR CHINTHAGINJALA received the M.Tech. degree in digital electronics and communication systems from Jawaharlal Nehru Technological University Anantapur, India, and the Ph.D. degree in communication networks from Vellore Institute of Technology, Vellore, India, in 2018. He is currently an Associate Professor with the School of Electronics Engineering, Department of Embedded Technology, Vellore Institute of Technology. His current research interests include communication networks, machine learning, deep learning, wireless sensor networks, and information security.

PRIYA NATHA received the B.Tech. degree in computer science and engineering from Jawaharlal Nehru Technological University Hyderabad, India, the M.Tech. degree in computer science and engineering from Osmania University, Hyderabad. She is currently an Assistant Professor with the Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India. Her research interests include artificial intelligence and machine learning, deep learning, data science, data mining, and software engineering.

GIOVANNI PAU (Senior Member, IEEE) received the bachelor's degree in telematics engineering from the University of Catania, Italy, and the master's (cum laude) and Ph.D. degrees in telematics engineering from the Kore University of Enna, Italy. He is currently an Associate Professor with the Faculty of Engineering and Architecture, Kore University of Enna. He is the author/co-author of more than 80 refereed papers published in journals and conference proceedings. His research interests include wireless sensor networks, fuzzy logic controllers, and intelligent transportation systems. He is a member of the IEEE (Italy Section) and has been involved in several international conferences as the session co-chair and a technical program committee member. He serves/served as a leading guest editor for the special issues of several international journals. He is an Editorial Board Member and an Associate Editor of several journals, such as IEEE Access, *Wireless Networks* (Springer), *EURASIP Journal on Wireless Communications and Networking* (Springer), *Wireless Communications and Mobile Computing* (Hindawi), *Sensors* (MDPI), and *Future Internet* (MDPI).

C. DHANAMJAYULU (Senior Member, IEEE) received the B.Tech. degree in electronics and communication engineering from JNTU University, Hyderabad, India, the M.Tech. degree in control and instrumentation systems from the Indian Institute of Technology Madras, Chennai, India, and the Ph.D. degree in power electronics from the Vellore Institute of Technology, Vellore, India. He was invited as a Visiting Researcher with the Department of Energy Technology, Aalborg University, Esbjerg, Denmark, funded by the Danida Mobility Grant, Ministry of Foreign Affairs of Denmark on Denmark's International Development Cooperation. He was a Postdoctoral Fellow with the Department of Energy Technology, Aalborg University, from October 2019 to January 2021. He is currently working as an Associate Professor with the Vellore Institute of Technology. His research interests include machine learning, federated learning, soft computing, computer vision, block chain, multilevel inverters, power converters, active power filters, power quality, grid connected systems, smart grid, and electric vehicle. He is an Academic Editor of the *journal International Transactions on Electrical Energy Systems*. He is also an Academic Editor of the *journal Mathematical Problems in Engineering*, an Academic Editor of the *PLOS ONE journal*, an Editorial Member for the *Scientific Reports Journal*, an Associate Editor for the *HardwareX Journal*, an Associate Editor for the *Heliyon Journal*, and an Associate Editor for the *Eprime Journal*.

FARUQ MOHAMMAD received the Ph.D. degree from the Department of Environmental Toxicology, Southern University and A&M College (Southern University), Baton Rouge, LA, USA, in May 2011, and the master's degree in organic chemistry from India. He is currently an Associate Professor of chemistry with the Surfactant Research Chair, Department of Chemistry, King Saud University, Saudi Arabia. His Postdoctoral Research experiences are from the Institute of Advanced Technology, Universiti Putra Malaysia, Malaysia; the School of Pharmacy, Northwest University, South Africa; and the Health Research Center, Southern University, USA. He served as a Lecturer for about three years in chemistry at Southern University. So far, he has published more than 190 journal research articles, five reviews, 17 book chapters, and three books edited. His research interests include the design and development of theranostic probes for the simultaneous diagnosis and therapy of cancer, hyperthermia-based therapy, polymeric nano drug delivery systems for malaria and cancerous diseases, understanding of the nanomaterials toxicity, natural biomaterials for sustainability, computational chemistry, and simulations related protein-drug interactions. He served as an Editor for *Austin Journal of Environmental Toxicology*. He is an Active Reviewer for *Journal of Colloidal and Interface Science* (Elsevier), *Materials Science and Engineering B* (Elsevier), *Nanoscale* (Royal Society of Chemistry), *Environmental Science: Nano* (Royal Society of Chemistry), and several MDPI journals, such as *Coatings*, *Materials*, *Polymers*, and *Pharmaceutics*.

Open Access funding provided by 'Università degli Studi di Enna KORE' within the CRUI CARE Agreement