Convolutional Neural Network-based Successive Cancellation List Decoder for the Polar Code

Sunil Yadav Kshirsagar, *Member, IAENG* and Venkatrajam Marka

***Abstract*—In modern communication systems, polar codes play an essential role as error-correcting codes due to their inclusion in 5G technology. Successive cancellation list (SCL) decoders with cyclic redundancy checks (CRCs) enhance polar code performance, but their computational complexity and latency are high. To overcome this limitation, this study proposes a convolutional neural network (CNN)-based SCL (CNN-based SCL) decoder. The proposed decoder significantly improves the error correction capability and decoding speed in terms of bit error rate (BER), block error rate (BLER), and frame error rate (FER) versus signal-to-noise ratio (SNR). For $L = 32$, the gained BER, BLER, and FER, respectively, are 2.34×10^{-6} at 3.5 dB SNR, 0.005, and 0.002 at 3 dB SNR. We conducted a simulation of the proposed decoder and compared the results with conventional decoders. The results indicate that the proposed CNN-based SCL decoder outperforms other decoders across various list sizes.**

***Index Terms*—Polar Code, Convolutional Neural Network (CNN), Successive Cancellation List (SCL) decoding, Bit Error Rate (BER), Block Error Rate (BLER), Frame Error Rate (FER).**

I. Introduction

ARIKAN introduced the concept of channel polarization to construct polar codes, which achieve the capacity for symmetric binary-input discrete memoryless channels (B-DMCs) by synthesizing highly reliable and unreliable channels through recursive transformations. This enables low-complexity successive cancellation (SC) decoding [1]. To enhance decoding performance, Tal and Vardy developed a successive cancellation list (SCL) decoder that maintains multiple decoding paths and selects the most probable path. The addition of cyclic redundancy check (CRC) precoding further improves reliability, making polar codes competitive with low-density parity-check (LDPC) codes [2]. Similarly, Elkelesh et al. proposed a belief propagation list (BPL) decoder that employs multiple belief propagation (BP) decoders on permuted factor graphs. This approach achieves a performance comparable to that of SCL, while enabling soft-output capability and lower latency [3]. To optimize polar codes for CRC-aided successive cancellation list (CA-SCL) decoding, Liao et al. utilized graph neural networks (GNNs) to construct polar codes using an iterative message-passing (IMP) algorithm, that efficiently minimizes frame error rates (FER) and scales across different block lengths [4]. Liu et al. introduced neural-network-assisted decoding schemes, including key-bit-based and last-subcode NN-assisted decoding. These approaches enhance error correction while maintaining low complexity, achieving superior bit error rate (BER) performance compared to conventional SC decoders [5].

Several studies have explored neural-network-assisted decoding approaches. A fully connected neural network has been proposed to reduce the decoding latency by leveraging parallel processing [6]. In contrast, convolutional neural networks (CNNs) improve BP decoding under colored noise conditions [7]. A sparse neural network (SNN) decoder optimizes non-binary polar codes, reducing complexity while maintaining decoding performance [8]. CNN-based BP flip decoding has demonstrated improved erroneous bit identification, thereby enhancing the decoding reliability for short codes [9]. Deep feed-forward neural networks have also been investigated for decoding polar codes, showing the potential for competitive error rates compared to list decoding [10]. In addition, deep learning-assisted SCL decoding with shifted pruning was introduced to balance the performance and complexity [11]. Fast SC decoding benefits from deep learning by partitioning the decoder into sub-blocks, thereby improving the computational efficiency [12]. A recurrent neural network (RNN)-based decoder with weight quantization reduces memory overhead and computational complexity while maintaining decoding accuracy [13]. A neural-network-aided path-splitting strategy optimizes SCL decoding by selectively splitting paths, thereby reducing decoding complexity without performance loss [14].

Recent advances in neural architectures have dramatically enhanced sensing, inference, and personalization tasks across diverse domains. In IoT security, an Edge Multi-Head GAT-GraphSAGE model augmented with FGSM adversarial training significantly improved intrusion detection on the NF-BoT-IoT dataset [15]. In computer vision, the YOLOv9-c framework—enhanced by Dynamic Snake Convolution and a Spatial-Channel Synergistic attention module—achieved an average precision of 0.902 for small infrared targets on the HIT-UAV dataset [16]. For graph-structured data, SMix-GNN employs k-reciprocal nearest neighbor graphs and S-Mixup augmentation to boost F1-scores by up to 5.63% across real-world classification tasks [17]. In digital signal processing, LSTM-based filter design adaptively learns coefficients via gating mechanisms to minimize amplitude error more effectively than traditional neural and windowed methods [18], while Environmental Transformers leverage multi-head attention and environment-aware feature extraction to improve non-line-of-sight signal classification in satellite navigation [19]. Beyond sensing, CNNs paired with MobileNetV2 transfer learning have been used to classify vegetable images at 95.78% accuracy for personalized vegetarian recipe recommendations, demonstrating neural models' versatility in real-world applications [20].

These breakthroughs in adaptive learning and attention

Manuscript received April 10, 2025; revised July 22, 2025.

Sunil Yadav Kshirsagar is a research scholar at the Department of Mathematics, School of Advanced Sciences, VIT-AP University, Amaravati, Andhra Pradesh, 522241, India (e-mail: sunilksagar143@gmail.com).

Venkatrajam Marka is an Associate Professor at the Department of Mathematics, School of Advanced Sciences, VIT-AP University, Amaravati, Andhra Pradesh, 522241, India (e-mail: mvraaz.nitw@gmail.com).

mechanisms have, in turn, inspired robust control strategies that mirror perception-driven architectures. Dynamic output quantization combined with persistent dwell-time switching has produced provably stable, fault-tolerant switched controllers under actuator faults echoing the gated updates of LSTM filters and graph attention networks [21]. In memristive neural networks, memoryless state-feedback controllers enable finite-time anti-synchronization, showcasing how neuromorphic dynamics can drive rapid convergence in networked systems [22]. Extending these ideas to fractional-order memristive systems, adaptive protocols guarantee predefined settling times independent of initial states, underscoring the power of complex-order dynamics for precise timing control [23]. In switched stochastic pure-feedback systems with incomplete measurements, an integrated approach combining state estimation, dynamic surface control, and RBF network approximation maintains semiglobally uniformly ultimately bounded performance despite packet loss and input saturation [24].

Deep learning-assisted polar-coded integrated data and energy networking (IDEN) was introduced to optimize both information and energy transmission, outperforming traditional model-based designs [25]. Neural-network-based adaptive polar coding dynamically adjusts to channel conditions and quality-of-service (QoS) requirements, outperforming 5G polar codes under SCL decoding [26]. Neural successive cancellation (NSC) decoding integrates neural networks with SC decoding, reducing latency while maintaining strong error correction performance [27]. Kshirsagar et. al. propose a hybrid polar code construction method that integrates a recurrent neural network for noise estimation with a Bald Hawk Optimization algorithm, achieving exceptionally low error rates [28]. Additionally, a CNN-aided bit-flipping mechanism enhances BP decoding by dynamically identifying erroneous bits and reducing the block error rate (BLER) while maintaining a low decoding latency [29]. A CNN-aided tree-based bit-flipping framework using imitation learning further optimizes BP-based decoding by introducing multi-bit flipping strategies, significantly improving error correction and reducing the number of flipping attempts [30]. Another CNN-based polar decoding approach enhanced decoding accuracy while maintaining computational efficiency, particularly for longer code lengths [31].

In this study, we developed a novel CNN-based SCL decoder to improve decoding capability and accuracy. The proposed approach integrates a CNN architecture with a conventional SCL decoder to reduce computational complexity and processing time. By training the CNN model on large sets of encoded messages and their noisy counterparts, the proposed decoder significantly enhances the error correction and decoding speed. CNNs, renowned for their pattern recognition capabilities, offer a distinct advantage in identifying the underlying structures in encoded data, thereby facilitating a more effective error correction. The primary contribution of this study is the implementation of a CNN-based SCL decoder for polar code decoding. The CNN is trained offline, and a linear simulation is conducted to assess the specific decoding gain. The experimental results demonstrate that the CNN-based SCL decoder achieves notable improvements in the bit error rate (BER), block error rate (BLER), and frame error rate (FER) performance compared to traditional SCL decoding methods.

The remainder of this paper is organized as follows. Section 2 provides an overview of the fundamental concepts of polar code and CNNs. Section 3 describes the proposed CNN-based SCL decoder. Section 4 presents a performance analysis of the proposed decoder and compares it with the existing methods. Section 5 summarizes the study and discusses potential future work.

II. Preliminaries

This section provides a comprehensive explanation of polar code encoding and decoding.

A. Polar Code

In 2009, Arikan made a revolutionary discovery of polar code, which was the first set of codes to achieve Shannon's capacity successfully. Polar encoding and decoding are crucial components of the polar code.

1) Polar Encoding: A polar code, denoted as $P(N, K)$, is a code with a code length of N and a coding rate of $R = K/N$. The equation that relates variables y and v is expressed as $y = vG_N$, where G_N represents the generator matrix for a given value of N. For $N = 2$, the specific value of G_2 is given by matrix, $G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

The generator matrix derived from the polar transform is typically employed to encode polar codes with a code length of N (where N is equal to 2^s, and s is greater than or equal to 1). The generator matrix, represented as G_N, is defined as follows:

$$G_N = T^{\otimes s}$$

where $T = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $T^{\otimes s}$ refers to the s^{th} Kronecker power of T. Encoding is the process of obtaining the encoded bits $y = (y_1, y_2, ..., y_N)$ by multiplying a specified source vector $v = (v_1, v_2, ..., v_N)$ by the matrix G_N, that is, $y = vG_N$. The vector $v = (v_1, v_2, ..., v_N)$ represents the encoded bit sequence. The bits should be reversed to perform proper encoding. The encoding bits $(v_1, v_2, ..., v_N)$ possess diverse levels of reliability as mandated by the polar design concept. M bits are divided into two subsets, each representing a different reliability level, to assess overall reliability appropriately. Information transmission uses only the K most reliable bits, whereas the remaining $N - K$ bits are set to zero.

2) Successive Cancellation List Decoding: The SC decoder sequentially decodes the received bits and makes decisions based on the previously decoded bits. Although the SC decoder is efficient, it has performance constraints, particularly when it comes to short block lengths. The SCL decoder maintains a record of potential codewords, thereby enhancing the likelihood of identifying an accurate codeword. The decoder generates numerous decoding paths, which are then removed based on their probabilities. The decoder commences with an initial inventory of paths, each with a probability of one. As the decoding process continues, the list is updated to include additional potential code words. Every time, the decoder adds a possible value (0 or 1) to the list of paths for the current bit, considering both extremes.

Consequently, the number of potential routes was doubled. The decoder eliminates paths according to the probability of controlling the list size. Up to a specific size, L, it can hold the most likely paths. The log-likelihood ratio (LLR) can be used as a measure to determine the probability of each path. Using this metric, we can find pathways with a better chance of matching the transmitted codeword. After processing all the bits, the decoder selects the path from the list that most likely represents the decoded codeword.

In polar codes, the Successive Cancellation List (SCL) decoder refines the SC decoder. SCL decoders increase error-correction performance by accounting for multiple decoding paths, as opposed to SC decoders. For large list sizes, the performance of the SCL decoder approaches that of the maximum likelihood (ML) decoder, resulting in higher performance. To strike a balance between computational complexity and efficiency, the value of L can be modified to represent the list size. Increased list sizes provide enhanced performance, but can also lead to affordable complexity.

III. Methodology

A. Motivation

Polar codes are widely recognized for their capacity-achieving performance during Successive Cancellation (SC) decoding. However, SC decoding suffers from significant limitations, particularly at finite block lengths, where incorrect early-stage decisions lead to error propagation. In addition, SC decoding lacks reliability in high-noise environments, resulting in increased BER. To address these issues, SCL decoding was introduced, which maintains multiple decoding paths and utilizes a Cyclic Redundancy Check (CRC) to select the most likely decoded sequence. Although SCL decoding improves performance by retaining multiple decoding candidates, it still suffers from drawbacks such as higher computational complexity and suboptimal path selection, especially in challenging noise conditions where CRC-based decisions may not always be reliable.

To further enhance error correction, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have emerged as promising solutions. CNNs have demonstrated remarkable success in pattern recognition and sequence-based tasks, owing to their ability to extract hierarchical features. In the context of decoding, CNNs offer key advantages, such as pattern extraction from error-prone sequences, robustness against noise, and soft decision refinement. Unlike traditional hard decision-based methods, CNNs can recognize complex error patterns and improve the reliability of the decoded sequences. This motivates the integration of CNNs with SCL decoding to refine decoding decisions and enhance the performance in noisy environments.

The proposed CNN-based SCL Decoder introduces a CNN as a postprocessing step after SCL decoding. Initially, the SCL decoder generates multiple candidate codewords and selects the most likely sequence using the CRC verification. However, because the CRC-based selection may not always be optimal, the CNN further evaluates the best candidate and verifies its accuracy before producing the final output. This approach ensures that the decoder retains the benefits of SCL decoding while leveraging CNNs to correct CRC failures and enhance decoding accuracy. By incorporating deep learning into the decoding process, the model achieves improved error correction, robustness to noise, and better path selection efficiency than conventional SCL decoders.

B. Architecture of CNN-based SCL decoder

Figure 1 illustrates an advanced decoding architecture that employs a combination of traditional Successive Cancellation List (SCL) decoding with a Convolutional Neural Network (CNN) to decode polar-coded messages. The decoding approach integrates classical signal processing methods with deep learning to significantly enhance the decoding performance, especially in noisy channel conditions. Initially, the input message bits are encoded via a Polar Encoder. Polar coding exploits the phenomenon of channel polarization, effectively transforming a set of noisy channels into subsets of highly reliable and unreliable channels, thereby facilitating robust error-correction capabilities. After encoding, these bits are modulated using binary phase-shift keying (BPSK), preparing them for transmission across a physical communication channel. The modulated signals then pass through an Additive White Gaussian Noise (AWGN) channel, which introduces realistic noise conditions to simulate real-world scenarios that the decoder must effectively handle. Following transmission, noisy signals are received by the combined SCL and CNN decoder. In this stage, the classical SCL decoder first produces initial estimates of the transmitted bits by exploring multiple decoding paths simultaneously, providing preliminary decoded bits along with reliability measures (soft decisions). However, SCL decoding alone may not adequately capture the complex error patterns introduced by the channel, thereby motivating the inclusion of a CNN-based decoder to complement and refine estimates.

The CNN-based decoder comprises several convolutional and dense layers arranged systematically to capture intricate patterns in the noisy signals. Specifically, the decoder includes three convolutional layers (Conv2D) with decreasing numbers of filters: 128, 64, and 32. Each convolutional layer extracts hierarchical spatial features from the input, enabling the decoder to identify subtle patterns that traditional decoding can overlook. These layers are each followed by Rectified Linear Unit (ReLU) activation functions to introduce nonlinearity and dropout layers for regularization, thereby reducing the risk of overfitting. After feature extraction, a flattened layer reshapes the multidimensional feature maps into a one-dimensional feature vector, preparing data for processing using subsequent dense (fully connected) layers. The decoder then employs three Dense layers. The first dense layer comprised 128 neurons, followed by a second layer of 64 neurons, each accompanied by ReLU activation and dropout regularization to enhance generalization further. Finally, the last dense layer outputs the decoded information bits using a sigmoid activation function, which converts the output values into probabilities between 0 and 1, thereby providing binary decision thresholds for accurate decoding.

The final output represents the estimated decoded bits that closely match the original transmitted information. By leveraging the strengths of both classical SCL decoding and modern deep learning techniques, this CNN-based hybrid decoding approach achieves superior decoding accuracy and

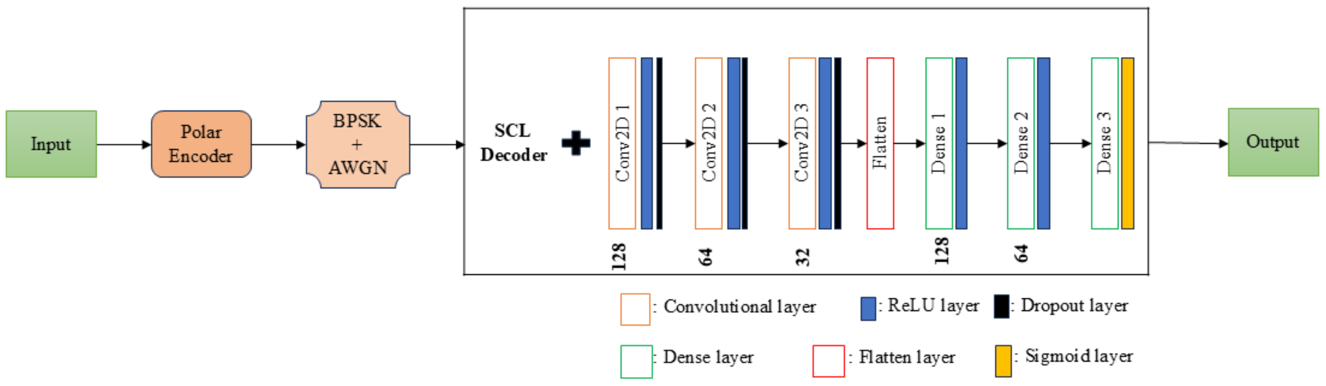


Fig. 1: Architecture of the proposed CNN-based SCL decoder

robustness, effectively reducing bit error rates, even under challenging noise conditions, making it particularly suitable for advanced polar-coded communication systems.

C. CNN-based SCL Decoder

CNNs are a type of sophisticated neural network that is used for the analysis of visual information. They have demonstrated remarkable efficacy in tasks such as image recognition, object detection, and natural language processing. CNNs are specially developed to autonomously and flexibly acquire organizational structures and characteristics from the input images. They consist of several layers: convolutional, pooling, and connected layers. The convolutional layer serves as the fundamental component of the CNN. This process entails the convolution of a series of filters, sometimes known as kernels, with input. An activation function is used to integrate the nonlinear characteristics into the model following the convolution process. The ReLU is the most commonly used activation function. Pooling layers were used to decrease the spatial dimensions (width and height) of the feature maps while preserving the most significant information. The predominant type is max pooling. Fully connected layers resemble conventional neural network layers, in which each neuron establishes connections with every neuron in the preceding layer.

The CNN-based SCL decoders includes the following steps.

1) Inputs: In the decoding process, several key inputs are utilized to ensure accurate codeword recovery. Channel log-likelihood ratios (LLRs), denoted as $\ell \in \mathbb{R}^N$, provide reliable information for each received bit. The bit-type indicator, $\mathbf{b} \in \{0,1,2\}^N$, classifies bits into information bits (0), shortened bits (2), and frozen bits (other values). Set $\mathcal{I}$ specifies the indices of the information bits essential for correctly extracting the transmitted message. Additionally, k_{crc} represents the number of bits allocated for the CRC check, whereas L defines the list size used in the SCL decoder to enhance decoding reliability. CRC verification relies on a predefined polynomial, CRC_{poly}, to ensure error detection in decoded messages. To further refine the decoding accuracy, a convolutional neural network (CNN) instantiated via the CNN evaluates the predicted codeword and decides whether to accept or reject it. If the CNN confirms the decoded sequence, the final decoded codeword $\mathbf{s}$ is returned; otherwise, the output is `none`, indicating a decoding failure. This hybrid approach leveraging SCL decoding and CNN-based verification enhances robustness, particularly in high-noise scenarios.

2) SCL Decoder Process: SCL decoding begins with initializing several crucial components that facilitate the decoding process. First, the input log-likelihood ratios (LLRs) representing the reliability values of the received bits are provided. The length N of these input LLRs is determined and the parameter $n = \log_2(N)$ is computed accordingly. For each decoding path $d = 1, 2, \ldots, L$, where L represents the decoder's list size, the algorithm initializes three main data structures: the log-likelihood ratio matrix $LLR^{(d)} \in \mathbb{R}^{(n+1)\times 2^n}$, decision matrix $s^{(d)} \in \{-1,0,1\}^{(n+1)\times 2^n}$, and path metric $PM^{(d)}$. Specifically, $LLR^{(d)}[n,:]$ is initialized with the input LLR values, the decision matrix is initialized with -1, and the path metrics are initialized as $PM^{(1)} = 0$ and $PM^{(d)} = \infty$ for $d > 1$. SCL decoder includes the following steps:

(i) Bit-by-Bit Decoding Procedure: Each bit index i undergoes decoding based on its type classification: information, shortened, or frozen bits. This classification determines the decoder's computational strategy for each bit position.

- Information bits: The LLR is computed, the decision is set to 0, and the path metric is updated using Equations 1 and 2, as follows:

$$LLR^{(d)}[0,i] = L_i(0, i, LLR^{(d)}, s^{(d)}) \tag{1}$$

$$PM^{(d)} = PM^{(d)} - min\{LLR^{(d)}[0,i] < 0\} \tag{2}$$

- Shortened bits: LLR computation is similar to information bits, however, decisions are directly set to zero without altering the path metrics.
- Frozen bits: The LLR computation is followed by decision based on the sign of LLR, and the distance metric (DM) is updated using Equation 3 and 4 as follows:

$$s^{(d)}[0,i] = \begin{cases} 1, & \text{if } LLR^{(d)}[0,i] < 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$DM^{(d)} = |LLR^{(d)}[0,i]| \tag{4}$$

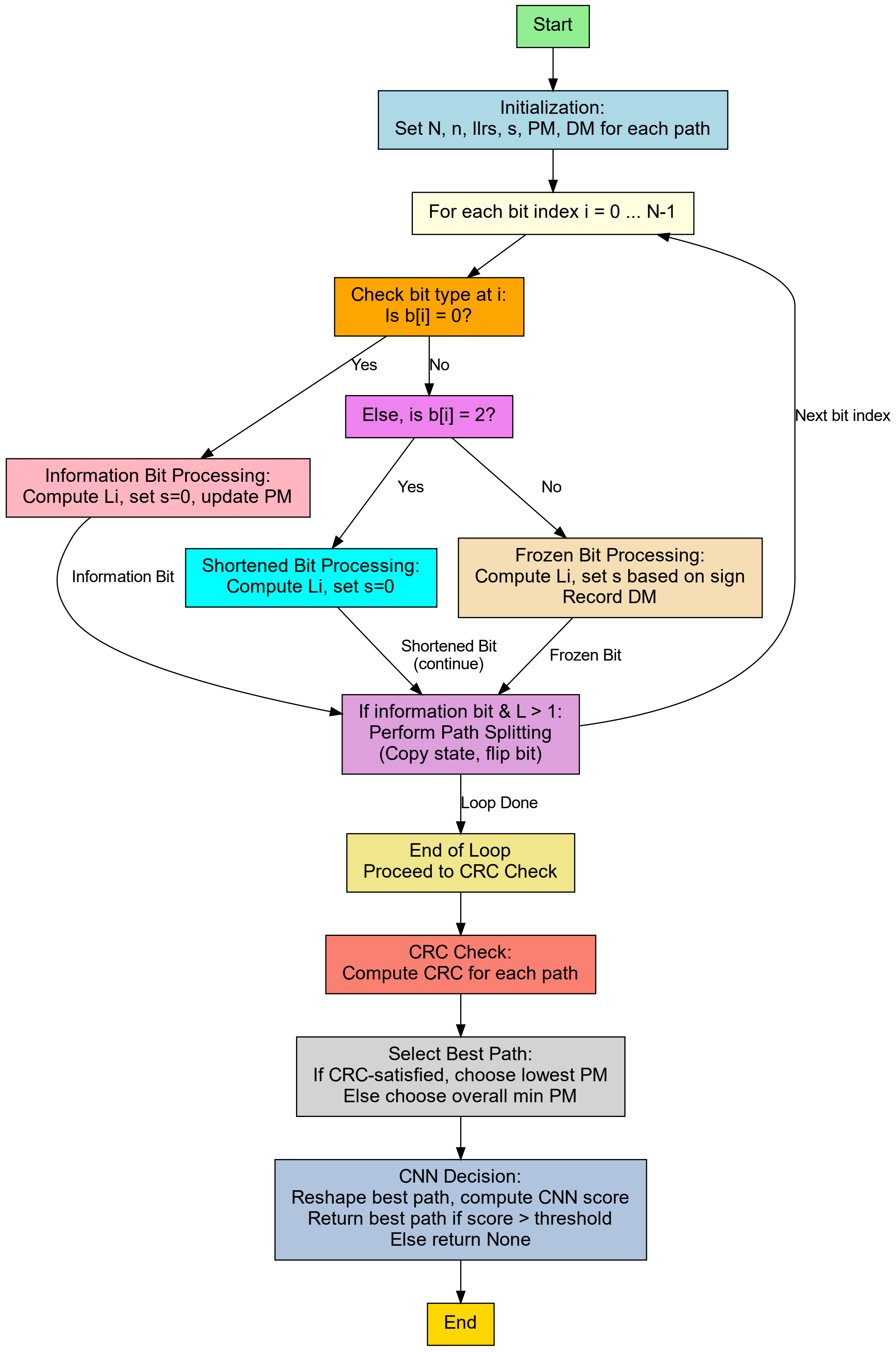

Start
Initialization: Set N, n, llrs, s, PM, DM for each path
For each bit index i = 0 ... N-1
Check bit type at i: Is b[i] = 0?
Yes
No
Else, is b[i] = 2?
Next bit index
Information Bit Processing: Compute Li, set s=0, update PM
Yes
No
Shortened Bit Processing: Compute Li, set s=0
Frozen Bit Processing: Compute Li, set s based on sign Record DM
Information Bit
Shortened Bit (continue)
Frozen Bit
If information bit & L > 1: Perform Path Splitting (Copy state, flip bit)
Loop Done
End of Loop Proceed to CRC Check
CRC Check: Compute CRC for each path
Select Best Path: If CRC-satisfied, choose lowest PM Else choose overall min PM
CNN Decision: Reshape best path, compute CNN score Return best path if score > threshold Else return None
End

Fig. 2: Flowchart of the proposed CNN-based SCL decoder

(ii) Path splitting and selection: For information bits, a combined metric vector is constructed and sorted, facilitating the selection of the top L paths with the smallest metrics. Path states are copied and the corresponding decisions are flipped at bit index i, updating the path metrics accordingly.

3) CRC Check and Path Selection: After generating multiple candidate decoding paths via successive cancellation list (SCL) decoding, a Cyclic Redundancy Check (CRC) is applied to verify the correctness of the candidate paths. For each candidate decoding path d, CRC verification is conducts using Equation 5, as follows:

$$crc_{iter}^{(d)} = CRC_{calc}\left(s^{(d)}[0, I[1:k_{crc}]], CRC_{poly}\right) \quad (5)$$

where, $CRC_{calc}(\cdot)$ denotes the CRC computation using the predefined polynomial CRC_{poly}. The candidate path is considered to be CRC-satisfied if

$$crc_{iter}^{(d)} = s^{(d)}[0, I[k_{crc}+1:]] \quad (6)$$

The path selection stage considers three possible scenarios.

- Single CRC-satisfied path: If only one decoding path passes the CRC check, the decision is straightforward. This unique path is directly chosen as the most reliable candidate, and decoding moves forward with this selection.
- Multiple CRC-satisfied paths: Among the multiple CRC-satisfied paths, the decoder selects the path with the minimal path metric ($PM^{(d)}$) using Equation 7, which represents the highest reliability:

$$s^* = \arg\min\{PM^{(d)} \mid crc_{iter}^{(d)} \text{ satisfied}\} \quad (7)$$

- **No CRC-satisfied paths:** In the challenging scenario where none of the candidate paths pass the CRC verification, the decoder again defaults to select the path that exhibits the minimum path metric among all possibilities using Equation 8, despite CRC failure, as it represents the ”least unreliable” decoding outcome:

$$s^* = \arg\min\{PM^{(d)} \mid d = 1, 2, \ldots, L\} \quad (8)$$

This careful combination of CRC verification and path metric analysis ensures that the decoder maintains a robust balance between error detection accuracy and decoding reliability, effectively mitigating decoding errors and reducing the likelihood of accepting erroneous messages.

4) Final CNN-Based Decision: Despite rigorous CRC check and path selection procedures, residual decoding errors may persist, especially under high-noise conditions or when CRC fails to discriminate effectively. To address this issue and further enhance decoding robustness, a sophisticated final verification step is introduces by employing a CNN. Once the best decoding path s^* is determined through the previous steps, the candidate codeword is prepared specifically for the CNN-based verification. The selected path s^* is reshaped into a format compatible with CNN processing and represented mathematically using Equation 9, as follows:

$$cnn_{input} = reshape\,(s^*) \quad (9)$$

The CNN, which is trained extensively on correct and incorrect decoded messages under various channel conditions, evaluates this reshaped candidate codeword to generate a prediction score using Equation 10,

$$score = CNN.predict(cnn_{input}) \quad (10)$$

This prediction score, which typically ranges from 0 to 1, quantifies the confidence of the CNN regarding the correctness of the candidate codeword. A higher score indicates strong confidence in correctness, whereas a lower score indicates potential decoding errors. A predefined threshold (commonly 0.5) is used to interpret this prediction clearly using Equation 11,

$$\text{Decoded Output} = \begin{cases} s^*, & \text{if } score > 0.5, \\ \text{None}, & \text{otherwise} \end{cases} \quad (11)$$

By integrating this CNN-based verification into the final decision stage, the decoder effectively leverages the pattern recognition capabilities of deep learning. This significantly enhances its resilience against residual decoding errors that traditional CRC checks alone may not detect, particularly in complex, high-noise scenarios. Thus, the CNN-enhanced decision framework fundamentally strengthens the overall reliability, robustness, and effectiveness of the CNN-based SCL decoding methodology.

The pseudocode and flowchart of the CNN-based SCL decoder are provided in Algorithms 1 and figure 2, respectively.

Algorithm 1 CNN based SCL Decoder

```
Require: $\ell \in \mathbb{R}^N$                         ▷ LLR channel values
Require: $\mathbf{b} \in \{0,1,2\}^N$   ▷ Bit type: 0 for information, 2 for shortened, else frozen
Require: $\mathcal{I}$                                  ▷ Indices of information bits
Require: $k_{crc}$                                      ▷ Number of CRC bits
Require: $L$                                            ▷ List size for SCL
Require: $\mathrm{CRC}_{poly}$                          ▷ CRC polynomial
Require: CNN model                              ▷ CNN with a predict function
Ensure: Decoded codeword $\mathbf{s}$ or None
 1: $N \leftarrow \mathrm{length}(\ell)$, $n \leftarrow \log_2(N)$
 2: for $d \leftarrow 1$ to $L$ do
 3:     Initialize $\mathrm{LLR}^{(d)} \in \mathbb{R}^{(n+1)\times 2^n}$ with $-\infty$
 4:     Set $\mathrm{LLR}^{(d)}[n,:] \leftarrow \ell$
 5:     Initialize $\mathbf{s}^{(d)} \in \{-1,0,1\}^{(n+1)\times 2^n}$ with $-1$
 6:     $\mathrm{PM}^{(d)} \leftarrow 0$ if $d = 1$, else $\infty$
 7:     $\mathrm{DM}^{(d)} \leftarrow 0$
 8: end for
 9: for $i \leftarrow 0$ to $N-1$ do
10:     if $\mathbf{b}[i] = 0$ then                       ▷ Information Bit
11:         for $d \leftarrow 1$ to $L$ do
12:             $\mathrm{LLR}^{(d)}[0,i] \leftarrow Li(0, i, \mathrm{LLR}^{(d)}, \mathbf{s}^{(d)})$
13:             $\mathbf{s}^{(d)}[0,i] \leftarrow 0$
14:             $\mathrm{PM}^{(d)} \leftarrow \mathrm{PM}^{(d)} - min\{\mathrm{LLR}^{(d)}[0,i] < 0\}$
15:         end for
16:     else if $\mathbf{b}[i] = 2$ then                  ▷ Shortened Bit
17:         for $d \leftarrow 1$ to $L$ do
18:             $\mathrm{LLR}^{(d)}[0,i] \leftarrow Li(0, i, \mathrm{LLR}^{(d)}, \mathbf{s}^{(d)})$
19:             $\mathbf{s}^{(d)}[0,i] \leftarrow 0$
20:         end for
21:     else                                         ▷ Frozen Bit
22:         for $d \leftarrow 1$ to $L$ do
```

```
23:             LLR$^{(d)}[0, i] \leftarrow Li(0, i, \text{LLR}^{(d)}, \mathbf{s}^{(d)})$
24:             if LLR$^{(d)}[0, i] < 0$ then
25:                 $\mathbf{s}^{(d)}[0, i] \leftarrow 1$
26:             else
27:                 $\mathbf{s}^{(d)}[0, i] \leftarrow 0$
28:             end if
29:             DM$^{(d)} \leftarrow \left|\text{LLR}^{(d)}[0, i]\right|$
30:         end for
31:     end if
32:     if $\mathrm{b}[i] = 0$ and $L > 1$ then
33:         Form PM_DM $\leftarrow$ [PM$^{(1)}, \ldots,$ PM$^{(L)}$, PM$^{(1)}$ + DM$^{(1)}, \ldots,$ PM$^{(L)}$ + DM$^{(L)}$]
34:         idx_sort $\leftarrow$ argsort(PM_DM)
35:         Define idx_min_low $\leftarrow \{i - L \mid i \in$ idx_sort$[1 \ldots L]$, $i \geq L\}$
36:         Define idx_min_up $\leftarrow \{i \mid i \in$ idx_sort$[L + 1 \ldots 2L]$, $i < L\}$
37:         for each pair $(d_{\text{low}}, d_{\text{up}})$ in corresponding indices do
38:             LLR$^{(d_{\text{up}})} \leftarrow$ copy(LLR$^{(d_{\text{low}})}$)
39:             $\mathbf{s}^{(d_{\text{up}})} \leftarrow$ copy($\mathbf{s}^{(d_{\text{low}})}$)
40:             $\mathbf{s}^{(d_{\text{up}})}[0, i] \leftarrow 1 - \mathbf{s}^{(d_{\text{low}})}[0, i]$
41:             PM$^{(d_{\text{up}})} \leftarrow$ PM_DM$[d_{\text{low}} + L]$
42:         end for
43:     end if
44: end for
45: for $d \leftarrow 1$ to $L$ do
46:     crc_iter$^{(d)} \leftarrow$ CRC_calculator$\Big(\mathbf{s}^{(d)}[0, \mathcal{I}[1 : k_{crc}]]$, CRC$_{poly}\Big)$
47:     Mark path $d$ as CRC-satisfied if
            $$\text{crc\_iter}^{(d)} = \mathbf{s}^{(d)}[0, \mathcal{I}[k_{crc} + 1 : ]$$
48: end for
49: if exactly one path is CRC-satisfied then
50:     $\mathbf{s}^* \leftarrow$ the unique CRC-satisfied path
51: else if multiple paths are CRC-satisfied then
52:     $\mathbf{s}^* \leftarrow$ argmin{ PM$^{(d)}$ : $d$ is CRC-satisfied}
53: else
54:     $\mathbf{s}^* \leftarrow$ argmin{ PM$^{(d)}$ : $1 \leq d \leq L$}
55: end if
56: if $\mathbf{s}^* \neq$ None then
57:     cnn_input $\leftarrow$ reshape($\mathbf{s}^*, (1, -1, 1)$)
58:     score $\leftarrow$ CNN.predict(cnn_input)
59:     if score $> 0.5$ then
60:         return $\mathbf{s}^*$
61:     else
62:         return None
63:     end if
64: else
65:     return None
66: end if
```

IV. Results and Discussion

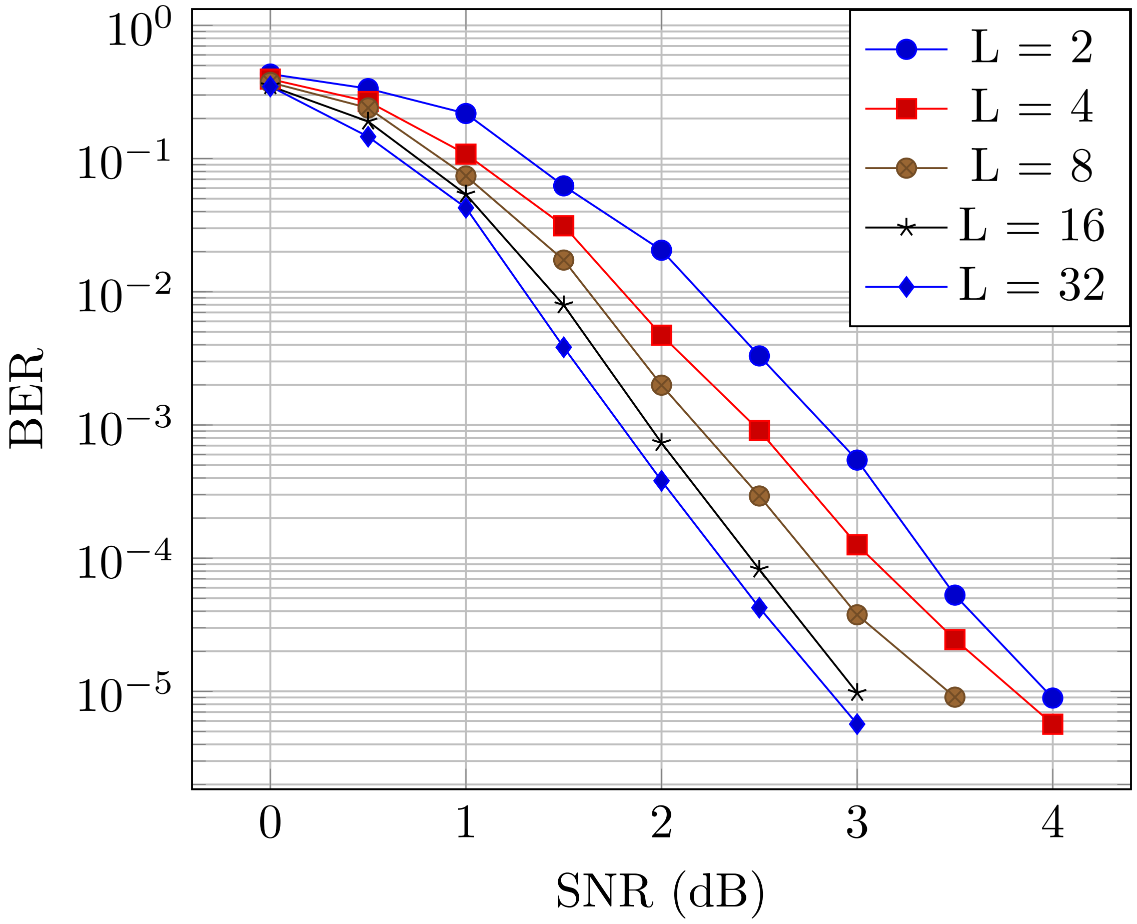
In this section, we describe the performance analysis for various list sizes using a CNN-based SCL decoder with $(1024, 512)$ polar code. Moreover, we compared the results of the CNN-based SCL decoder with those of other existing techniques. We conducted our experiment on a Windows 10 system with 64 GB of RAM, using Python programming in a high-performance computing (HPC) lab.

A. Performance Analysis based on List Sizes

Figure 3(a) illustrates the BER performance of $(1024, 512)$ Polar Code decoded using a CNN-based SCL decoder. The decoder is tested with various list sizes, specifically $L = 2, 4, 8, 16,$ and 32. We assessed the performance across a spectrum of SNR values ranging from 0 dB to 4 dB. As the SNR increases, the BER decreases for all list sizes, as expected. A higher SNR indicates better signal quality, leading to fewer errors in the decoded information, and expanding the size of list L results in improved BER performance, particularly at higher SNRs. However, this improvement is accompanied by an increase in computing complexity and memory requirements. Increasing the list size requires more computer resources, which results in a slower and more resource-intensive decoding procedure. The choice of list size should depend on the specific requirements of the application, balancing the desired error performance with available computational resources. For systems with strict BER requirements, employing a larger list size, such as $L = 32$, is preferable because it significantly reduces the error rate. However, for systems with tight resource restrictions, a balance must be achieved between the targeted bit BER performance and the available computational resources. Under such circumstances, an intermediate list size, such as $L = 8$ or $L = 16$, may offer a fair compromise.

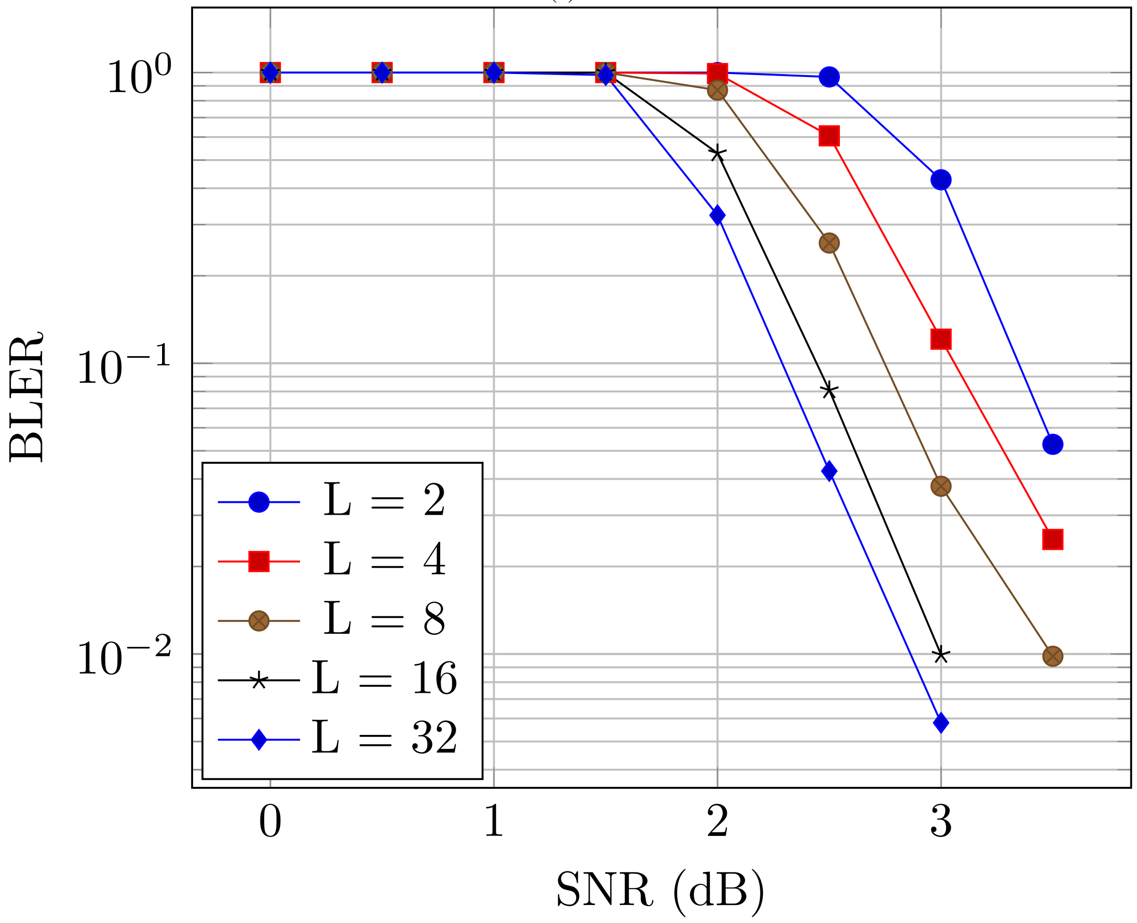
Figure 3(b) shows the BLER performance of $(1024, 512)$ Polar Code decoded using a CNN-based SCL decoder. As the SNR increases, the BLER decreases for all list sizes, which is expected because a higher SNR implies better signal quality and less noise, leading to an improved decoding performance. The performance improves significantly as the list size L increases. This is evident from the shift in the BLER curves to the left as L increases. Although increasing the list size improves the performance, the gain in the BLER starts to saturate at larger list sizes. For example, the difference in performance between $L = 16$ and $L = 32$ is less pronounce than that between $L = 2$ and $L = 4$. This suggests that beyond a certain point, increasing the list size yields diminishing returns in terms of BLER improvement. The CNN-based SCL decoder shows a significant performance improvement with increasing list size, as evidence by the lower BLER at each SNR level for larger values of L. However, the performance gains diminish at higher list sizes, indicating a point of diminishing returns, where further increases in L offer only a marginal improvement.

Figure 3(c) illustrates the FER performance of $(1024, 512)$ Polar Code decoded using a CNN-based SCL decoder. As the SNR increases, the FER decreases for all list sizes, which is expected because a higher SNR implies better signal quality and less noise, leading to an improved decoding performance. The performance improves significantly as the list size L



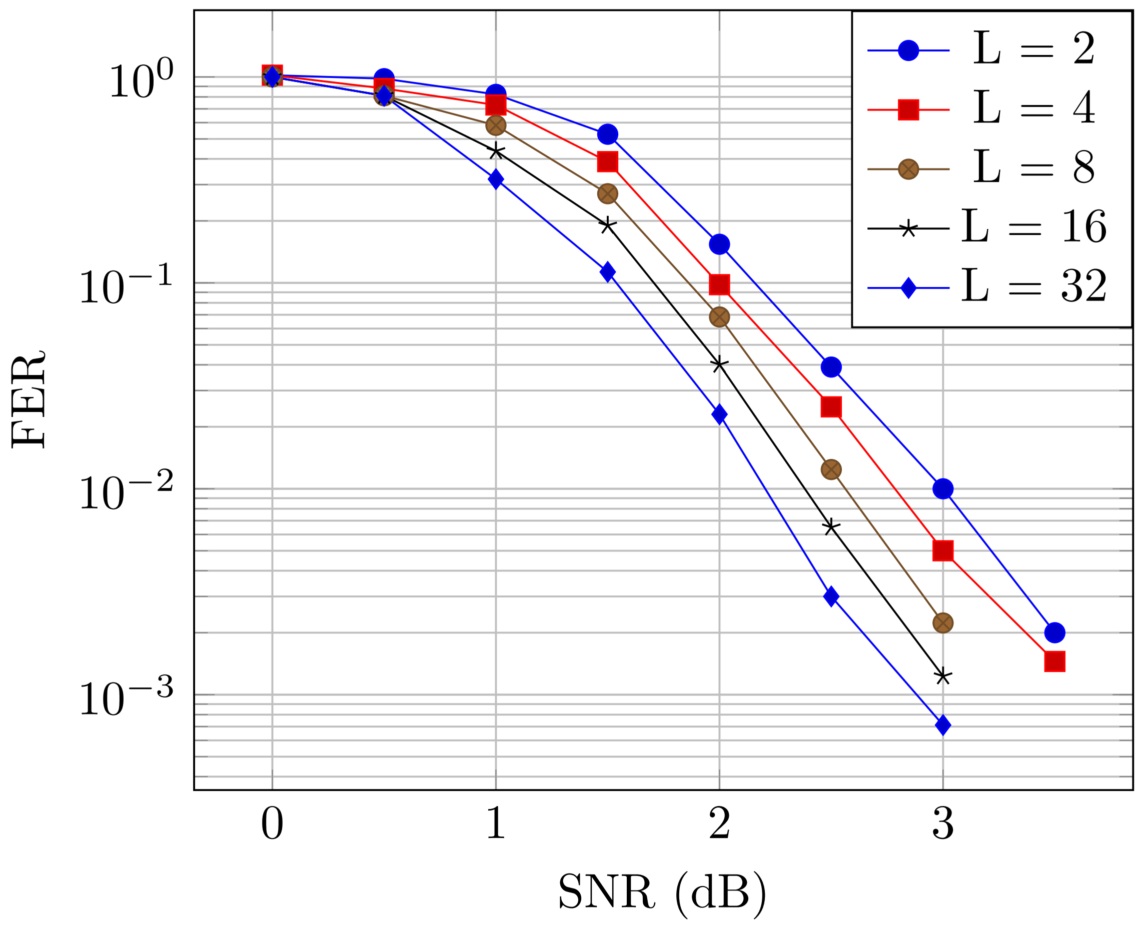
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
BER
0
1
2
3
4
SNR (dB)
L = 2
L = 4
L = 8
L = 16
L = 32

(a)



10^0
10^{-1}
10^{-2}
BLER
0
1
2
3
SNR (dB)
L = 2
L = 4
L = 8
L = 16
L = 32

(b)



(c)

Fig. 3: Performance analysis of the proposed CNN-based SCL decoder for the $(1024, 512)$ polar code for different list sizes

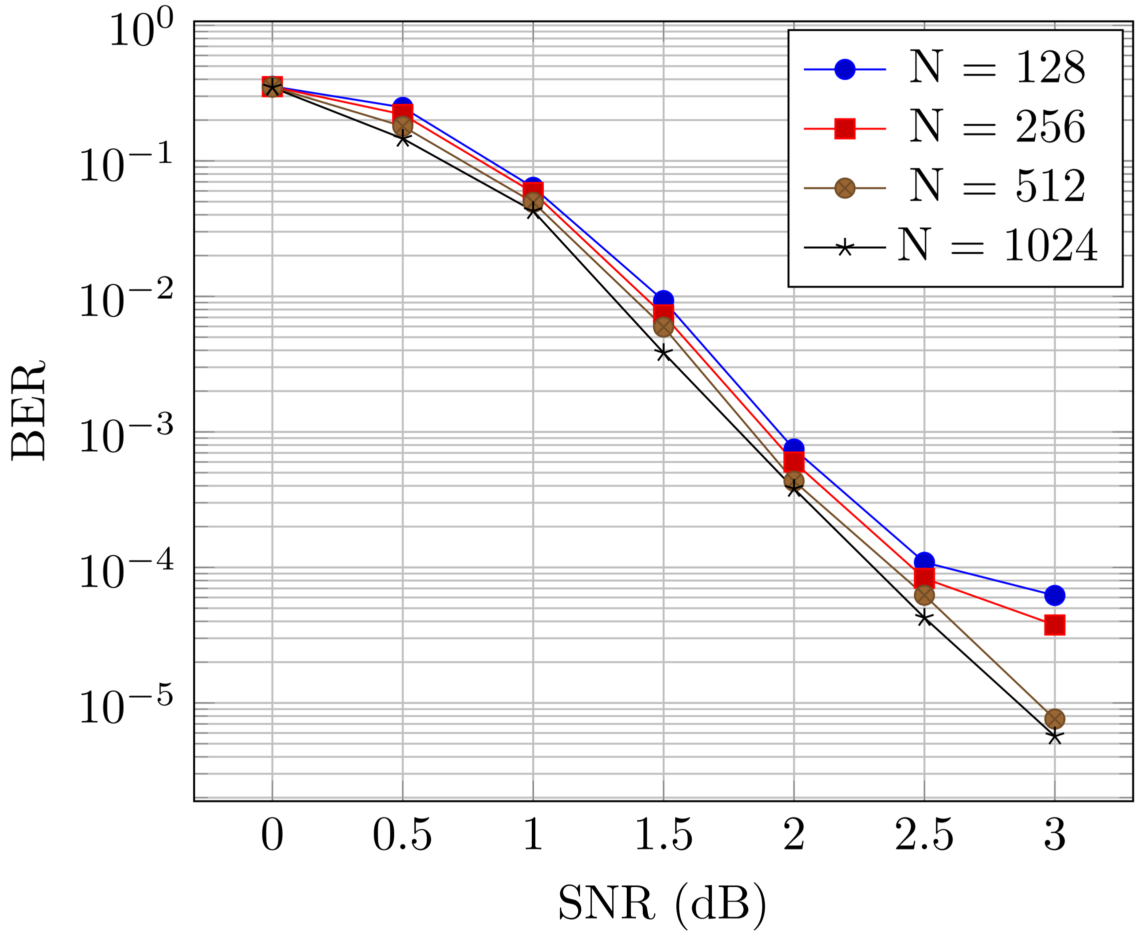
increases. This is evident from the shift in the FER curves to the left as L increases. Although increasing the list size improves the performance, the gain in the FER starts to saturate at larger list sizes. For example, the difference in performance between $L = 16$ and $L = 32$ is less pronounce than that between $L = 2$ and $L = 4$. This suggests that beyond a certain point, increasing the list size yields diminishing returns in terms of FER improvement. The CNN-based SCL decoder shows a significant performance improvement with increasing list size, as evidence by the lower FER at each SNR level for larger values of L. However, the performance gains diminish at higher list sizes, indicating a point of diminishing returns, where further increases in L offer only a marginal improvement.

B. *Performance Analysis based on Code Length*

The figure 4 (a) illustrates the BER in relation to the SNR measured in decibels for polar codes of varying block lengths N. As the SNR rises from 0 dB to 3 dB, the BER for all code lengths declines significantly, demonstrating that an elevated SNR results in a reduction of bit errors. At low SNR values, all four graphs demonstrate elevated BERs; however, as SNR rises, the BER for each block length declines significantly. Significantly, extended block lengths consistently yield reduced BERs at equivalent SNR levels compared to shorter lengths, indicating enhanced error-correction efficacy with larger N. For example, at around 2 dB, the $N = 1024$ curve achieves a BER near 10^{-4}, whereas $N = 128$ is closer to 10^{-3}, underscoring the benefits of bigger block sizes in enhancing reliability.

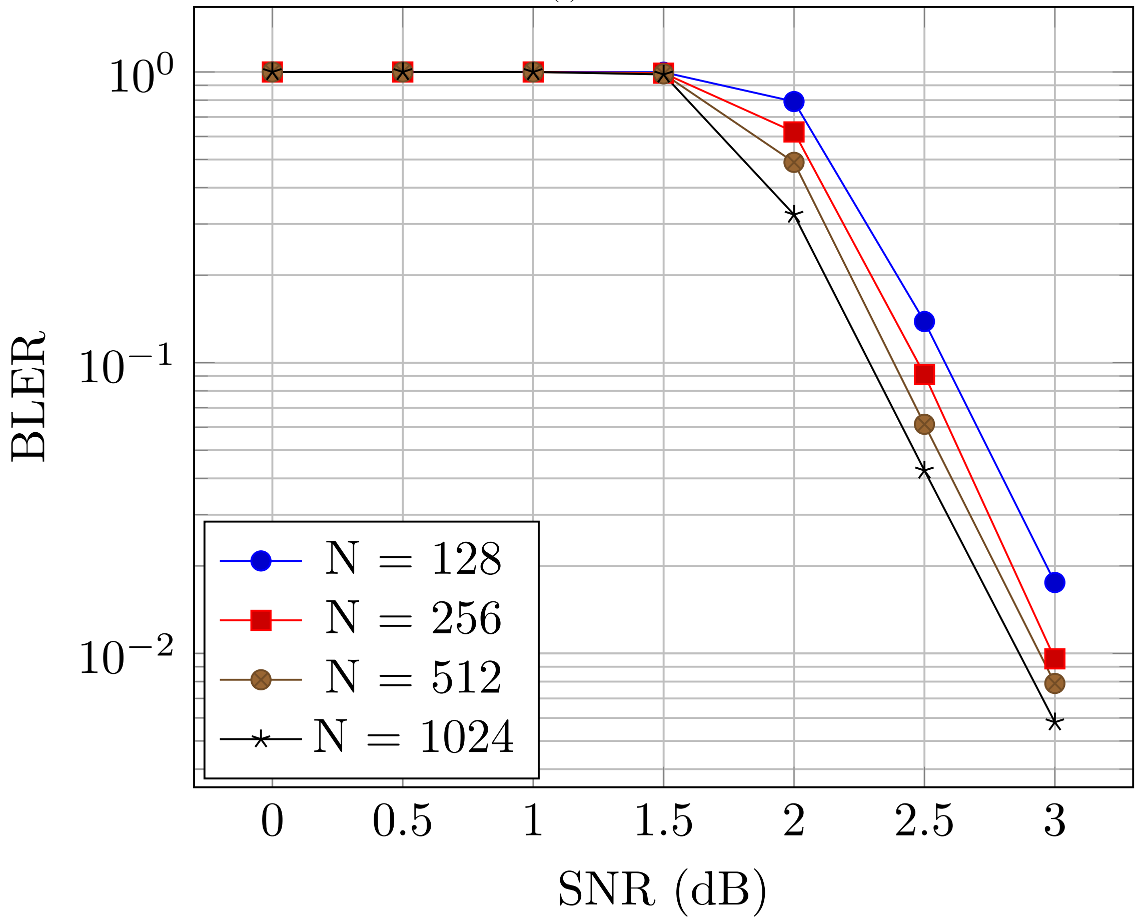
Figure 4 (b) shows how well polar codes perform in terms of BLER with different block lengths N when used over a noisy channel, based on the SNR measured in decibels. Each curve demonstrates that, for a fixed SNR, increasing the block length N yields a lower BLER, indicating stronger error-correcting capability. At low SNR values (around 0 dB to 1.5 dB), all code lengths exhibit relatively high BLERs equal to 1. As SNR increases beyond 1.5 dB, the BLER for longer codes (e.g., $N = 1024$) drops more rapidly, achieving values below 10^{-2} near 3 dB. Conversely, shorter codes like $N = 128$ sustain BLERs near 10^{-2} even at 3 dB, indicating the drawback of diminished length. This figure demonstrates that polar codes perform substantially better with extended block lengths, particularly at moderate to high SNR, where the error count is markedly diminished for large N.

Figure 4 (c) shows how well polar codes perform in terms of FER with different block lengths N when used over a noisy channel, based on the SNR measured in decibels. Each curve demonstrates that, for a fixed SNR, increasing the block length N yields a lower FER, indicating stronger error-correcting capability. At low SNR values (around 0 dB to 1 dB), all code lengths exhibit relatively high FERs. As SNR increases beyond 1 dB, the FER for longer codes (e.g., $N = 1024$) drops more rapidly, achieving values below 10^{-3}



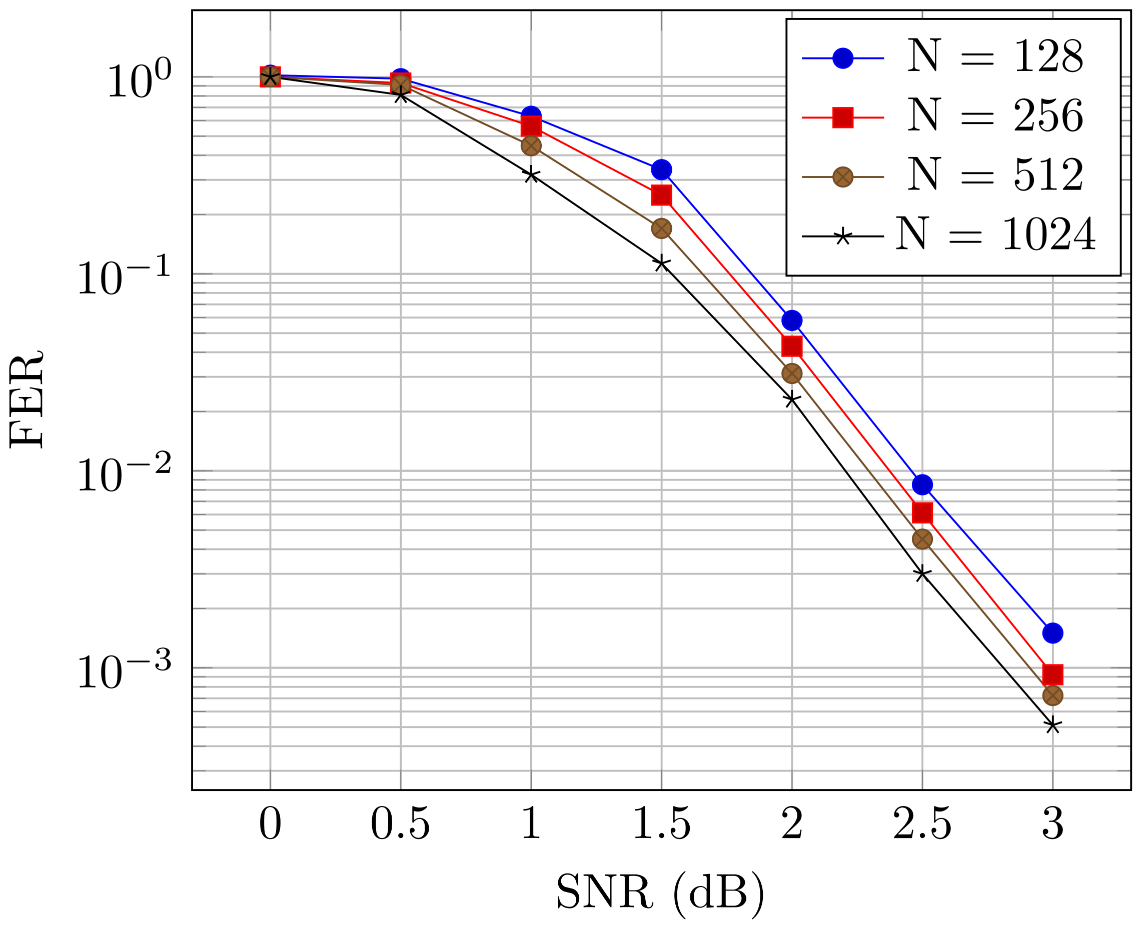
BER
SNR (dB)
N = 128
N = 256
N = 512
N = 1024
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
0.5
1
1.5
2
2.5
3

(a)



BLER
SNR (dB)
N = 128
N = 256
N = 512
N = 1024
10^0
10^{-1}
10^{-2}
0
0.5
1
1.5
2
2.5
3

(b)



(c)

Fig. 4: Performance analysis of the proposed CNN-based SCL decoder for the various code length with list size $L = 32$.

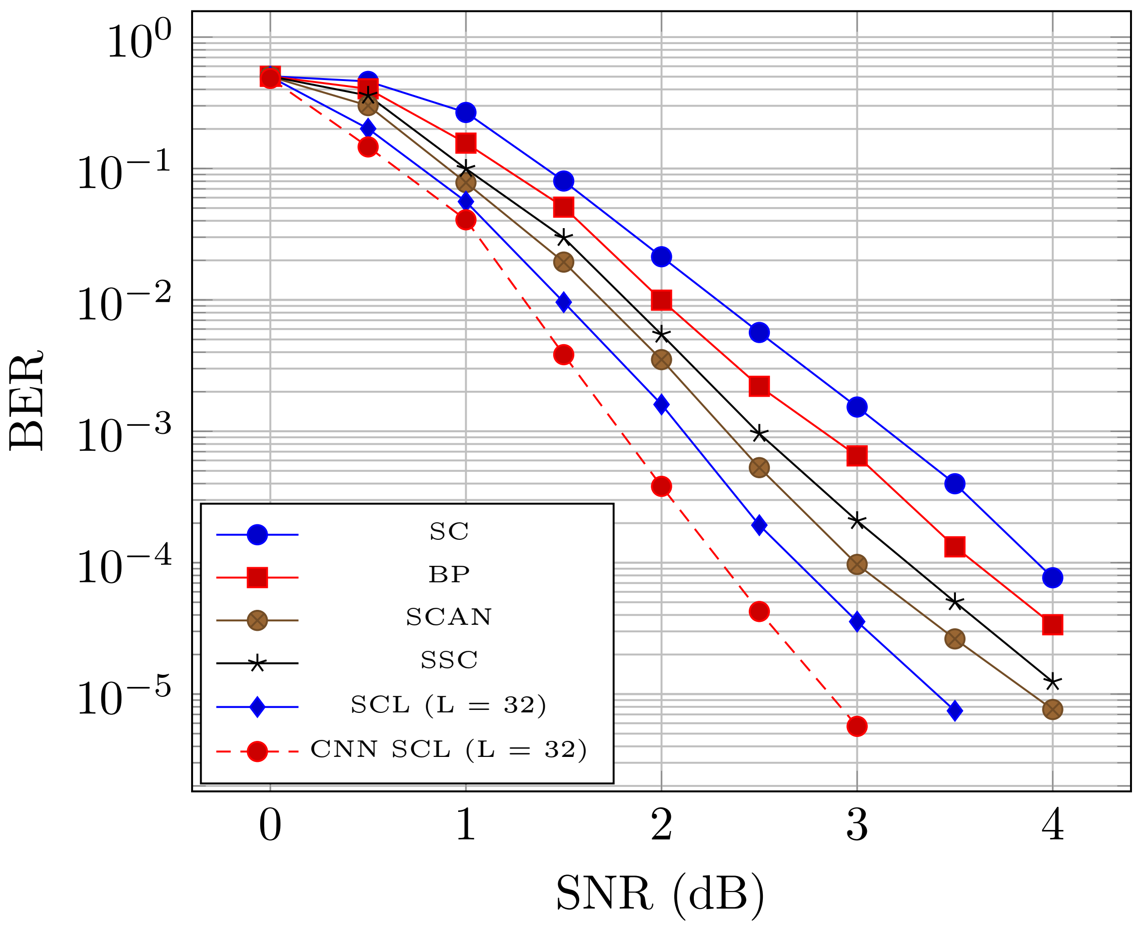
near 3 dB. Conversely, shorter codes like $N = 128$ sustain FERs near 10^{-2} even at 3 dB, indicating the drawback of diminished length. This figure demonstrates that polar codes perform substantially better with extended block lengths, particularly at moderate to high SNR, where the error count is markedly diminished for large N.

C. Comparative Analysis

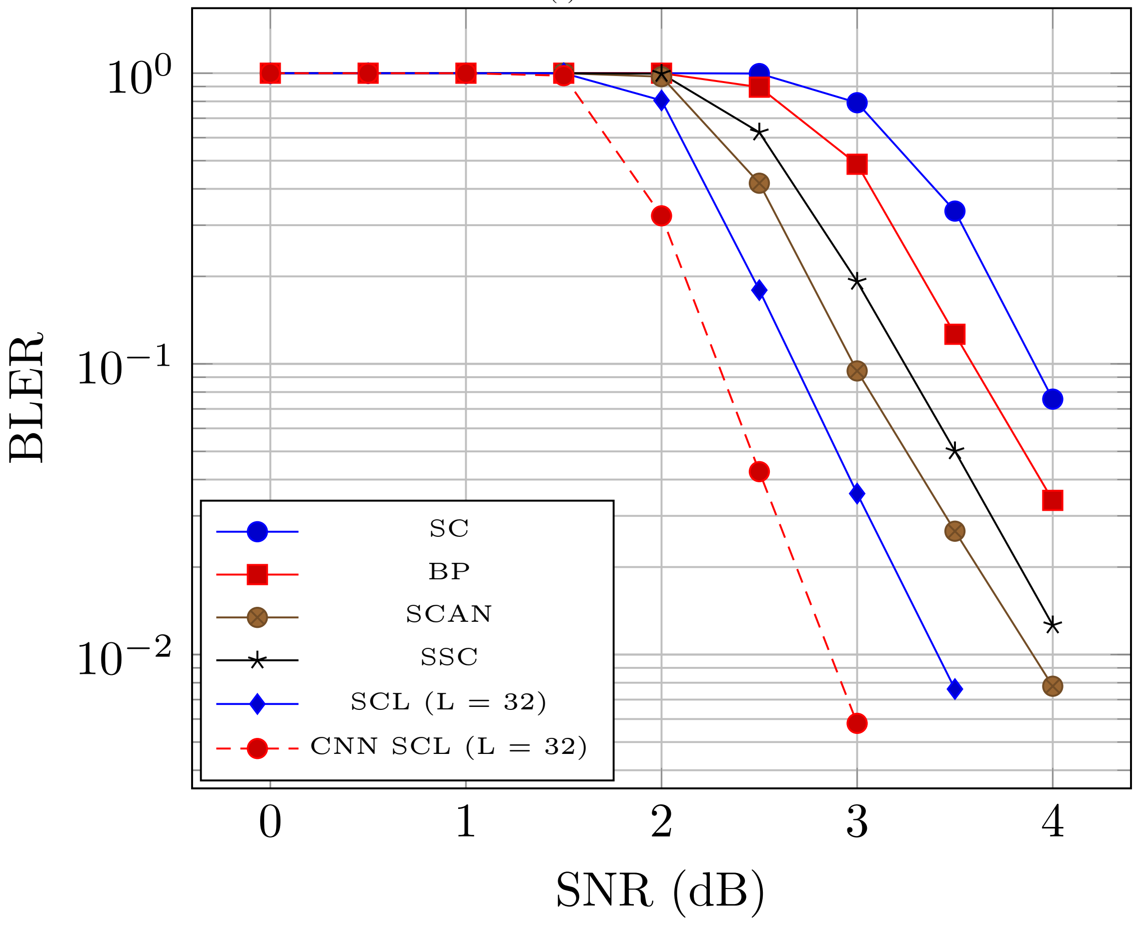
This section presents a detailed comparative analysis of the proposed CNN-based SCL decoder and existing decoders. We compared the CNN-based SCL decoder with SC[32], BP[33], SCAN[34], SSC[35], and SCL[36] (L = 32). Figure 5(a) presents a comparative analysis of the BER for different decoders versus the SNR for the $(1024, 512)$ Polar Code. This clearly shows that the proposed CNN-based SCL decoder with a list size of 32 outperforms all the other decoders. At 2 dB SNR, the CNN-based SCL decoder reaches BER levels below 10^{-3}, showing high efficiency in decoding under noise. However, at higher SNR values, the CNN-based SCL decoder achieves a BER as low as 10^{-6}, indicating excellent performance even in low-noise environments. The proposed CNN-based SCL decoder with a list size of 32 demonstrates the best BER performance across all SNRs, making it the most robust among the compared decoders, particularly in low-noise scenarios. This comparative analysis clearly indicates that advanced CNN-based SCL decoders offer substantial improvements in BER, especially as the SNR increases.

Figure 5(b) presents a comparative analysis of the BLER for different decoders versus the SNR for the $(1024, 512)$ Polar Code. This clearly shows that the proposed CNN-based SCL decoder provides the best BLER performance across all SNR values, outperforming all the other decoders. At $2.5\,dB$ SNR, the proposed CNN-based SCL decoder achieves an BLER below 10^{-1}, demonstrating superior noise resilience. The CNN-based SCL decoder, on the other hand, can reach BLER levels low than 10^{-2} when the SNR is higher. This shows that it is effective at fixing block errors, even in places with little noise. The proposed CNN-based SCL decoder with a list size of 32 achieves the lowest BLER across all SNR values, demonstrating its superior performance and making it the most effective for communication systems where low block error rates are critical. This comparative analysis demonstrates that an advanced CNN-based SCL decoder is highly effective in reducing BLER, especially at higher SNR levels. This decoder is best suited for applications where maintaining low BLER is critical for reliable communication.

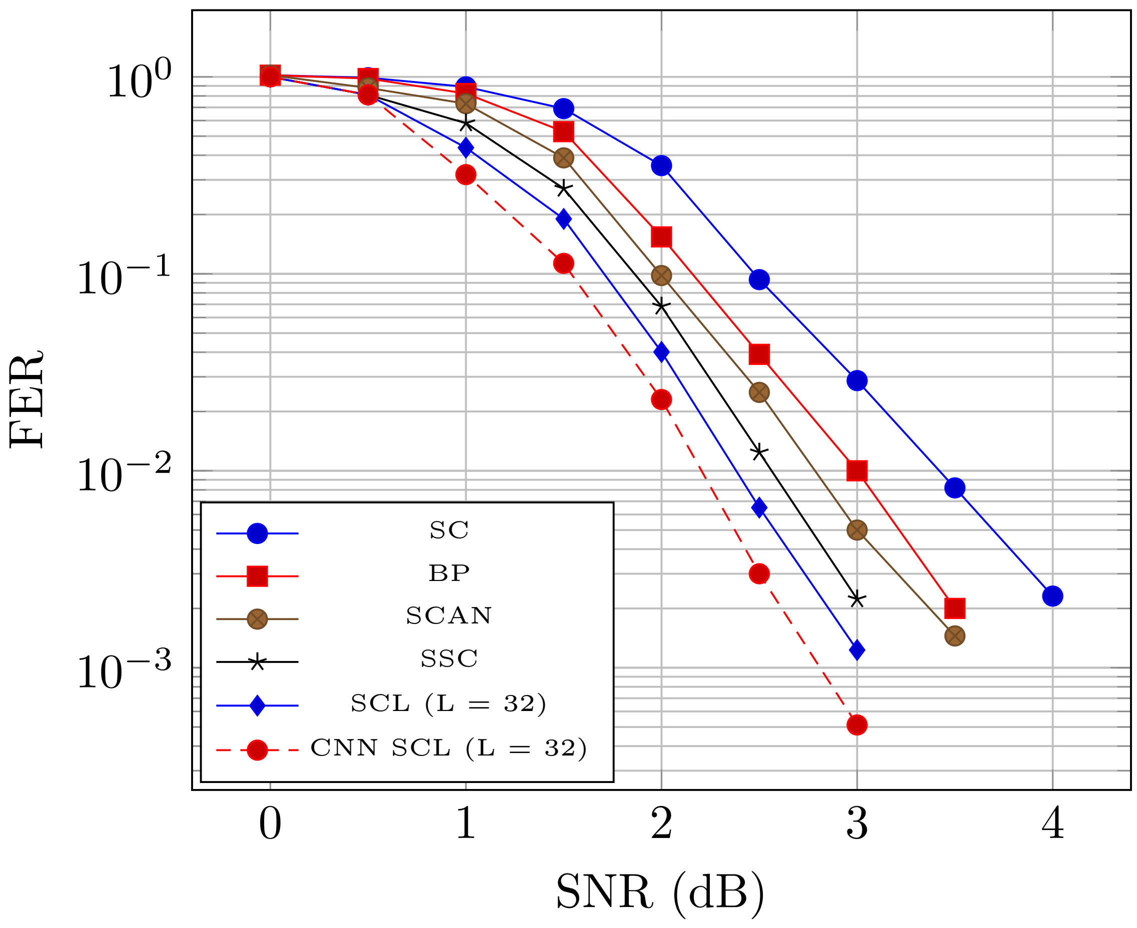
Figure 5(c) presents a comparative analysis of the FER for different decoders versus the SNR for the $(1024, 512)$ Polar Code. This clearly shows that the proposed CNN-based SCL decoder provides the best FER performance across all SNR values, outperforming all the other decoders. At $1.5\,dB$ SNR, the proposed CNN-based SCL decoder achieves an


BER
SNR (dB)
SC
BP
SCAN
SSC
SCL (L = 32)
CNN SCL (L = 32)

(a)


BLER
SNR (dB)
SC
BP
SCAN
SSC
SCL (L = 32)
CNN SCL (L = 32)

(b)



(c)

Fig. 5: Comparative analysis of the proposed CNN-based SCL decoder for the $(1024, 512)$ polar code and other existing decoders

FER below 10^{-1}, demonstrating superior noise resilience. The CNN-based SCL decoder, on the other hand, can reach FER levels as low as 10^{-3} when the SNR is higher. This shows that it is effective at fixing frame errors, even in places with little noise. The proposed CNN-based SCL decoder with a list size of 32 achieves the lowest FER across all SNR values, demonstrating its superior performance and making it the most effective for communication systems where low frame error rates are critical. This comparative analysis demonstrates that an advanced CNN-based SCL decoder is highly effective in reducing FER, especially at higher SNR levels. This decoder is best suited for applications where maintaining low frame error rates is critical for reliable communication.

TABLE I: BER analysis of proposed decoder with SC and SCL decoder for various code length

N	SNR (dB)	BER		
		SC	SCL ($L = 32$)	CNN-based SCL ($L = 32$)
128	1	0.786	0.0725	0.0638
	2	0.0864	0.00914	0.00074
	3	0.00794	0.000532	0.000062
256	1	0.547	0.0614	0.0584
	2	0.0736	0.0072	0.00059
	3	0.00674	0.000214	0.000037
512	1	0.354	0.0536	0.0494
	2	0.0437	0.00347	0.00043
	3	0.00398	0.000081	0.000007
1024	1	0.267	0.056	0.0407
	2	0.0213	0.0016	0.000381
	3	0.00153	0.000035	0.000005

D. Comparative Discussion

Table I compares BER analysis of three decoding methods: SC, SCL ($L = 32$), and the new CNN-based SCL decoder with $L = 32$, using various code lengths and SNR values. Increasing the SNR from 1 dB to 3 dB significantly lowers the BER for all decoders, regardless of the code length. The table demonstrates that, for all code lengths N, the classical SC decoder exhibits the highest BER at each SNR, while the SCL decoder ($L = 32$) consistently lowers the BER by roughly an order of magnitude. The CNN-enhanced SCL further reduces the BER, with the margin of improvement becoming more pronounced at higher SNRs. For instance, at SNR = 3 dB with $N = 1024$, BER falls from 3.5×10^{-5} (SCL) to 5×10^{-6} (CNN-SCL). As code length increases from 128 to 1024, all decoders experience a drop in BER at a given SNR; however, the relative gain from adding CNN layers remains substantial across lengths.

Table II compares BLER analysis of three decoding methods: SC, SCL ($L = 32$), and the new CNN-based SCL decoder with $L = 32$, using various code lengths and SNR values. Increasing the SNR from 1 dB to 3 dB significantly

TABLE II: BLER analysis of proposed decoder with SC and SCL decoder for various code length

N	SNR (dB)	BLER		
		SC	SCL ($L = 32$)	CNN-based SCL ($L = 32$)
128	1	1	1	1
	2	1	1	0.7912
	3	0.9248	0.10932	0.01753
256	1	1	1	1
	2	1	0.9567	0.6220
	3	0.8974	0.0814	0.00958
512	1	1	1	1
	2	1	0.8897	0.4887
	3	0.8514	0.0510	0.00788
1024	1	1	1	1
	2	1	0.8059	0.3230
	3	0.7915	0.0357	0.0057

lowers the BLER for all decoders, regardless of the code length. The SC decoder suffers the highest BLER, while SCL provides a marked reduction, e.g., for $N = 512$ at 2 dB, SCL FER is 0.0563 versus SC's 0.514. Introducing CNN layers yields further BLER gains; at SNR = 2 dB with $N = 256$, BLER drops from 0.0738 (SCL) to 0.0429 (CNN-SCL). Overall, CNN-SCL decoding has the lowest BLER in all situations, especially at moderate-to-high SNRs, where it cuts down frame errors significantly compared to traditional SCL.

TABLE III: FER analysis of proposed decoder with SC and SCL decoder for various code length

N	SNR (dB)	FER		
		SC	SCL ($L = 32$)	CNN-based SCL ($L = 32$)
128	1	0.92	0.875	0.631
	2	0.828	0.0924	0.058
	3	0.0805	0.00768	0.0015
256	1	0.915	0.742	0.562
	2	0.645	0.0738	0.0429
	3	0.0596	0.00578	0.00092
512	1	0.90	0.583	0.447
	2	0.514	0.0563	0.0312
	3	0.0414	0.00342	0.000723
1024	1	0.89	0.437	0.319
	2	0.3543	0.0401	0.023
	3	0.0287	0.00123	0.000512

Table III compares FER analysis of three decoding methods: SC, SCL ($L = 32$), and the new CNN-based SCL decoder with $L = 32$, using various code lengths and SNR values. Increasing the SNR from 1 dB to 3 dB significantly lowers the FER for all decoders, regardless of the code length. The SC decoder suffers the highest FER, while SCL provides a marked reduction, e.g., for $N = 512$ at 2 dB, SCL FER is 0.0563 versus SC's 0.514. Introducing CNN layers yields further FER gains; at SNR = 2 dB with $N = 256$, FER drops from 0.0738 (SCL) to 0.0429 (CNN-SCL). Overall, CNN-SCL decoding has the lowest frame error rate in all situations, especially at moderate-to-high signal-to-noise ratios, where it cuts down frame errors significantly compared to traditional SCL.

E. Computational Complexity Analysis

Figure 6 The figure illustrates the complexity analysis of various polar code decoders in terms of average decoding time (in milliseconds) versus code length. It compares the proposed CNN-based SCL decoder ($L = 32$) against traditional decoding methods including SC, BP, SCAN, SSC, and the conventional SCL ($L = 32$) decoder. As shown, the SC decoder demonstrates the lowest complexity across all code lengths, followed by SSC. While the traditional SCL decoder exhibits a steep increase in decoding time with larger code lengths due to its list-based processing, the CNN-based SCL decoder significantly reduces decoding time, especially for longer code lengths, maintaining competitive efficiency while preserving decoding performance. This highlights the advantage of leveraging convolutional neural networks to accelerate list decoding without compromising error-correction capability.

V. Conclusion

Decoders play a crucial role in the construction of polar code. Keeping this in mind, we presented an advanced CNN-based SCL decoder for polar code in this article. In terms of decoding, the proposed decoder outperformed traditional SCL decoders. The proposed approach makes use of the CNN's extraordinary feature extraction capabilities to improve the decoding process, ensure resistance against channel noise, and improve the error correction capability. Extensive simulations revealed that our CNN-based SCL decoder outperformed traditional SCL decoders in terms of BER, BLER and FER, even under challenging channel conditions. These findings imply that combining deep learning mechanisms with conventional decoding algorithms can significantly improve communication systems, allowing more effective and reliable data transmission. To make machine learning-based decoders work better and more useful in the real world, more research could be conducted on other neural network models and ways to make the network structure more efficient.

Author contributions statement

Dr. Venkatrajam Marka developed the project, which involved permission, visualization, supervision, and management.

Sunil Y. Kshirsagar created the approach, conducted the study, authored the paper, performed proofreading, and prepared the original manuscript.

All the authors have examined and approved the final version of the manuscript.

Data availability:

The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE transactions on information theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[3] A. Elkelesh, M. Ebada, S. Cammerer, and S. Ten Brink, "Belief propagation list decoding of polar codes," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1536–1539, 2018.

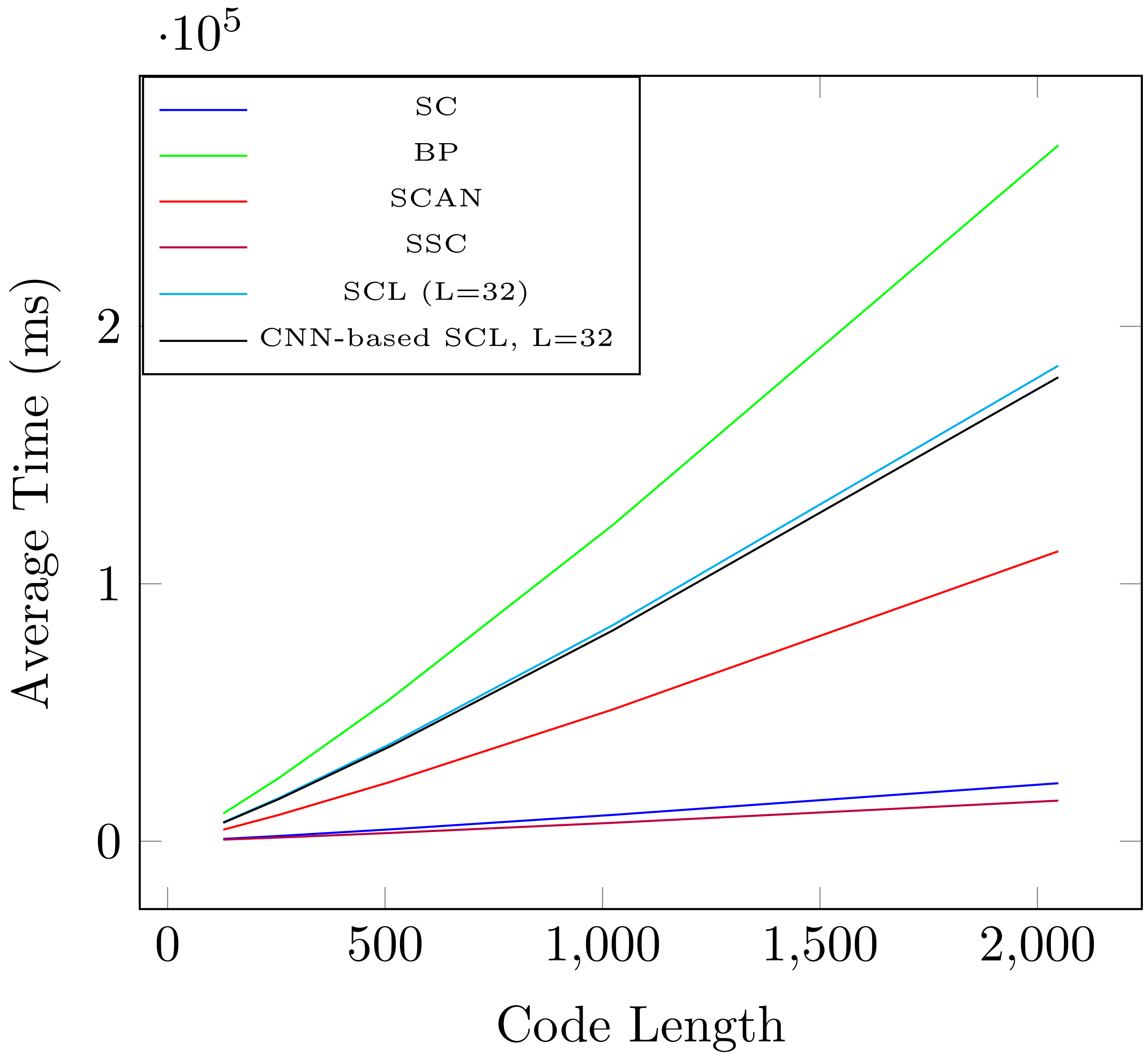


Fig. 6: Comparison of the computational complexity of the proposed CNN-based SCL decoder and other decoders

[4] Y. Liao, S. A. Hashemi, H. Yang, and J. M. Cioffi, “Scalable polar code construction for successive cancellation list decoding: A graph neural network-based approach,” *IEEE Transactions on Communications*, vol. 71, no. 11, pp. 6231–6245, 2023.

[5] H. Liu, L. Zhang, W. Yan, and Q. Ling, “Neural-network-assisted polar code decoding schemes,” *Applied Sciences*, vol. 12, no. 24, p. 12700, 2022.

[6] J. Á. Sánchez-Rodríguez, A. M. Martinez-Enriquez, and M. Lara, “A fully connected neural network for polar channel decoding,” in *2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE, 2023, pp. 1–6.

[7] C. Wen, J. Xiong, L. Gui, Z. Shi, and Y. Wang, “A novel decoding scheme for polar code using convolutional neural network,” in *2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2019, pp. 1–5.

[8] Y. Shu, H. Zhao, and C. Han, “A sparse neural network decoder for non-binary polar codes,” in *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2022, pp. 253–258.

[9] X. Zhang, Y. Qiu, W. Kong, J. Cui, and Y. Liu, “Bp flip decoding algorithm of polar code based on convolutional neural network,” in *2022 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. IEEE, 2022, pp. 444–449.

[10] J. Seo, J. Lee, and K. Kim, “Decoding of polar code by using deep feed-forward neural networks,” in *2018 international conference on computing, networking and communications (ICNC)*. IEEE, 2018, pp. 238–242.

[11] Y. Lu, M. Zhao, M. Lei, C. Wang, and M. Zhao, “Deep learning aided scl decoding of polar codes with shifted-pruning,” *China Communications*, vol. 20, no. 1, pp. 153–170, 2023.

[12] H. Feng, H. Xiao, S. Zhong, Z. Gao, T. Yuan, and Z. Quan, “Deep-learning-aided fast successive cancellation decoding of polar codes,” *Journal of Communications and Networks*, vol. 26, no. 6, pp. 593–602, 2024.

[13] C.-F. Teng, C.-H. D. Wu, A. K.-S. Ho, and A.-Y. A. Wu, “Low-complexity recurrent neural network-based polar decoder with weight quantization mechanism,” in *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2019, pp. 1413–1417.

[14] B. Dai, C. Gao, F. C. Lau, and Y. Zou, “Neural network aided path splitting strategy for polar successive cancellation list decoding,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9597–9601, 2023.

[15] B. Sun, H. Dai, J. Sun, and X. Wei, “Research on intrusion detection models with multi-head attention mechanism based on graph neural network.” *Engineering Letters*, vol. 33, no. 5, 2025.

[16] J. Zhang and Y. Zhang, “Infrared small target detection with uav based on convolutional neural networks.” *Engineering Letters*, vol. 33, no. 5, 2025.

[17] M. Rong, G. Zhang, X. Guo, Q. Sun, F. Hu, and H. Qi, “Smix-gnn: A

powerful graph neural network enhanced by aggregating graph mixup and k-reciprocal nearest neighbors." *Engineering Letters*, vol. 33, no. 5, 2025.

[18] F. Zhu, X. Yang, J. Yang, J. Yang, M. Zhang, and S. Liu, "Design of digital fir filter based on long short-term memory neural network." *Engineering Letters*, vol. 32, no. 10, 2024.

[19] Y. Zhao, Z. Dai, F. Li, X. Zhu, and C. Ran, "A new attention-based neural network for the identification of non-line-of-sight signals in data from global navigation satellite systems." *Engineering Letters*, vol. 32, no. 10, 2024.

[20] K. A. Ardisa, W. A. E. Prabowo, and S. Rustad, "Implementation of convolutional neural network method for detecting vegetables as recommendation for vegetarian food recipes," in *In Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science*, 2022, pp. 83–88.

[21] Y. Su, X. Wang, W. Tai, and J. Zhou, "Fault-tolerant quantized control for switched neural networks with actuator faults and dynamic output quantization," *IAENG International Journal of Applied Mathematics*, vol. 55, no. 1, pp. 7–15, 2025.

[22] L. Duan, Z. Zhang, and Z. Li, "Finite-time anti-synchronization for memristive neural networks with time-varying delays," *IAENG International Journal of Applied Mathematics*, vol. 55, no. 3, pp. 611–617, 2025.

[23] X. Feng and J. Gao, "Predefined-time synchronization of fractional-order memristive neural networks with time-varying delay," *IAENG International Journal of Applied Mathematics*, vol. 55, no. 4, pp. 763–767, 2025.

[24] Z. Li, L. Wang, H. Lv, and Z. Wang, "Adaptive neural network control for switched stochastic pure-feedback nonlinear systems with incomplete measurements," *IAENG International Journal of Applied Mathematics*, vol. 55, no. 5, pp. 1014–1027, 2025.

[25] L. Xiang, J. Cui, J. Hu, K. Yang, and L. Hanzo, "Polar coded integrated data and energy networking: A deep neural network assisted end-to-end design," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 11 047–11 052, 2023.

[26] V. Miloslavskaya, Y. Li, and B. Vucetic, "Neural network-based adaptive polar coding," *IEEE Transactions on Communications*, vol. 72, no. 4, pp. 1881–1894, 2023.

[27] N. Doan, S. A. Hashemi, and W. J. Gross, "Neural successive cancellation decoding of polar codes," in *2018 IEEE 19th international workshop on signal processing advances in wireless communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[28] S. Y. Kshirsagar and V. Marka, "Polar code construction by estimating noise using bald hawk optimized recurrent neural network model," *Scientific Reports*, vol. 15, no. 1, p. 23387, 2025.

[29] C.-F. Teng, A. K.-S. Ho, C.-H. D. Wu, S.-S. Wong, and A.-Y. A. Wu, "Convolutional neural network-aided bit-flipping for belief propagation decoding of polar codes," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7898–7902.

[30] C.-F. Teng and A.-Y. A. Wu, "Convolutional neural network-aided tree-based bit-flipping framework for polar decoder using imitation learning," *IEEE Transactions on Signal Processing*, vol. 69, pp. 300–313, 2020.

[31] Y. Qin and F. Liu, "Convolutional neural network-based polar decoding," in *2019 2nd World Symposium on Communication Engineering (WSCE)*. IEEE, 2019, pp. 189–194.

[32] C. Zhang, B. Yuan, and K. K. Parhi, "Reduced-latency sc polar decoder architectures," in *2012 IEEE International conference on communications (ICC)*. IEEE, 2012, pp. 3471–3475.

[33] Y. Yu, Z. Pan, N. Liu, and X. You, "Belief propagation bit-flip decoder for polar codes," *IEEE Access*, vol. 7, pp. 10 937–10 946, 2019.

[34] L. Zhang, Y. Sun, Y. Shen, W. Song, X. You, and C. Zhang, "Efficient fast-scan flip decoder for polar codes," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[35] P. Giard and A. Burg, "Fast-ssc-flip decoding of polar codes," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018, pp. 73–77.

[36] M.-C. Chiu and Y.-S. Su, "Design of polar codes and pac codes for scl decoding," *IEEE Transactions on Communications*, vol. 71, no. 5, pp. 2587–2601, 2023.