International Journal of Innovative Computing, Information and Control
Volume **22**, Number **3**, June **2026**

ICIC International ©2026 ISSN 1349-4198
pp. **767–782**

A LOW-PRECISION SUCCESSIVE CANCELLATION DECODING ALGORITHM FOR DEEP NEURAL NETWORK POLAR CODES

Jie Shen[1], Chang Yun[2], Wenfan Wang[3] and Guiping Li[2,*]

[1]School of Mechanical Engineering
North China University of Water Resources and Electric Power
No. 36, Beihuan Road, Zhengzhou 450045, P. R. China
shenjie@ncwu.edu.cn

[2]School of Computer Science and Technology
Xi'an Technological University
Jinhua Road, Xi'an 710021, P. R. China
llxghp@sina.com.cn; *Corresponding author: liguiping@xatu.edu.cn

[3]School of Information Engineering
Henan Vocational College of Water Conservancy and Environment
No. 136, Huayuan Road, Zhengzhou 450008, P. R. China
wwf@hwec.edu.cn

Received June 2025; revised October 2025

Abstract. *To enhance the error-correction capability of neural network decoders for polar codes and minimize the required memory overhead, this paper proposes a low-precision successive cancellation (SC) decoding algorithm. The algorithm is assisted by deep neural networks and incorporates quantization and partitioning techniques. Specifically, it quantizes the weights of the deep neural network decoder to a limited number of bits, utilizing an 8-bit quantized input and output in the Q8.4 fixed-point digital format for all arithmetic operations, thereby achieving int8 computations. With quantization-aware training, quantization errors are accounted for within the training loss, allowing the deep neural network to effectively learn how to decode with reduced precision. The proposed scheme, by employing fewer floating-point weights, reduces both memory overhead and the number of floating-point operations. Simulation results indicate that the proposed decoder decreases memory usage and decoding latency while maintaining a slightly improved bit error rate (BER) performance.*

Keywords: Polar codes, Successive cancellation, Weight quantization, Deep neural network, Partitioning

1. **Introduction.** Polar codes [1], introduced by Arikan, are celebrated for their outstanding analytical properties and can be applied in various scenarios such as the Internet of Things (IoT), edge computing, distributed AI training, and robot communications [2]. They have been designated as the control channel coding scheme for the fifth generation of enhanced Mobile Broadband (eMBB) due to their theoretical ability to reach the Shannon limit under the SC decoding algorithm. However, despite the low computational complexity of SC decoding, it is constrained by its serial nature, resulting in a decoding process that is slow in throughput and high in latency. This limitation is shared with SC List (SCL) decoding [3] and other improved SC versions [4]. As machine-type communication is expected to grow, the demands for ultra-reliable, low-latency, and low-power consumption short block codes are anticipated to become more stringent in 5G and 6G [5]. Although increasing the list size can improve the frame error rate (FER) of SCL-based

DOI: 10.24507/ijicic.22.03.767

decoding for short-block polar codes, it results in sub-optimal area scaling in hardware implementations.

There has been a growing interest in utilizing artificial intelligence (AI), particularly deep learning (DL), for wireless communication and channel encoding and decoding since [6] was published in 2016. Numerous studies have found that AI, and more specifically DL, has the potential to outperform traditional analytical methods in the design of communication systems [7, 8]. This is accomplished through the analysis of channel statistics, the extraction of features from wireless networks, and the identification of optimal communication algorithms – approaches that differ from traditional methods, which depend on mathematical models and expert knowledge. Since AI-driven approaches can fully capture real-world environments, they have demonstrated significant advantages in the design and optimization of wireless systems, including the joint optimization of encoding and decoding through end-to-end training [9].

Polar codes have been studied in conjunction with DL methods to assist conventional decoding algorithms, as first introduced in [10]. This approach utilized DL to train the edges of the Tanner graph for standard belief propagation (BP). Specifically, recurrent neural architectures, deep neural networks (DNNs), long short-term memory (LSTM) networks, and convolutional neural networks (CNNs) have been employed in SC decoding or its improved versions to enhance performance. For instance, the authors of [11] adopted the design concept of partitioned decoders and concatenated multiple neural networks with SC decoders to create a Neural SC (NSC) decoder, achieving equivalent decoding performance. This concept is akin to that of [10], which utilized several neural networks to substitute the sub-blocks of the BP decoder. In [12], a DNN is used to identify which bits to flip in BP flipping decoding. The authors in [13] explored how CNNs could reduce decoding latency. Since the aforementioned schemes are based on short blocks and are limited by the curse of dimensionality, [14] achieved scalability to arbitrary block lengths by employing a fully differentiable graph neural network (NN)-based architecture for channel decoding. Generally, decoding longer codes requires a larger neural network size. Such a network demands substantial computational resources during the training phase and imposes high computational and space complexities during the inference phase. In particular, the complexity during the inference phase is of practical importance, as training is typically conducted offline. To overcome the complexity challenges, LSTM neural networks have been applied to sequential decoding of polar codes due to their excellent sequence feature extraction ability and long short-term memory capabilities [15]. [16] achieved a significant reduction in complexity by using NNs to improve the flip efficiency of SCL decoding.

Generally, there are two prevalent strategies to mitigate the complexity introduced by deep learning methods in SC decoding. The first is neural network pruning, which aims to eliminate superfluous parameters from the original network while maintaining accuracy. The second strategy, parameter quantization, involves reducing the precision of neural network decoder parameters and decoding messages, including likelihood ratio (LLR) values derived from channel observations.

Although the BER of the pruned model may approach that of traditional SC decoding under high signal-to-noise ratios (SNRs), its performance significantly deteriorates in low SNRs or long-code scenarios. This is because the pruning operation removes redundant weight connections in the neural network, weakening its discriminative capability against channel noise. The pruned network, due to its simplified structure, exhibits reduced adaptability to channel variations. Particularly when the code length exceeds 64, pruning exacerbates the model's dependence on the training data distribution, making it difficult to generalize to new scenarios.

Mapping continuous LLR values to a finite set of discrete values inherently results in some loss of precision. In the context of neural network-based SC decoding of polar codes, this precision loss can impact the training and prediction performance of the neural network. To mitigate this precision loss, it may be necessary to enhance the complexity of the neural network, for instance, by incorporating additional neurons or layers, which in turn increases decoding complexity. Consequently, the LLR domain quantization scheme must consider the trade-off between accuracy and complexity.

Considering that the neural network decoder utilizes floating-point numbers for training and pretraining, which necessitates the storage of numerous weights, the associated floating-point operations are costly in terms of hardware and energy consumption. This leads to increased memory requirements and time overhead. Therefore, quantizing the neural network decoder parameters by converting floating-point parameters and activation values into low-precision integer representations can optimize the neural network. This approach significantly reduces the consumption of storage and computational resources. For SC decoding of polar codes based on neural networks, storing a large number of parameters as low-precision integers can substantially decrease the required storage space and reduce the time cost of decoding.

However, quantization inherently introduces errors during the process, which may result in reduced accuracy of the SC decoding for polar codes. To counteract these quantization errors, it is necessary to use specific optimization algorithms to fine-tune the quantization parameters, ensuring that the accuracy of the quantized model closely approximates that of the original model.

In this paper, we propose a low-precision SC decoder for deep neural networks, which incorporates quantization and partitioning techniques. The decoder is composed of multiple neural network layers connected in series with SC decoding. Our proposed scheme aims to reduce floating-point operations and the number of weights, thereby optimizing the complexity and performance of SC decoding within the context of deep learning. Simulation results indicate that the proposed low-precision SC decoding algorithm for deep neural networks achieves slight performance gains with reduced memory resource overhead and lower decoding latency.

The remainder of this paper is organized as follows. Section 2 reviews the polar codes, the original SC algorithm, and the neural channel decoding. Low-precision SC decoder based on DNN is proposed in Section 3. Section 4 analyzes the decoding performance, latency and weight storage. Conclusions are drawn in Section 5.

2. **Basic Theory.**

2.1. **Polar codes and successive cancellation decoding.**

Polar codes are a type of forward error correction coding method that relies on the polarization process. Through polarization transformation of the original channel, channels with a capacity approaching 1 are considered noise-free and perfect for transmitting information bits, while channels with a capacity approaching 0 are utilized for transmitting frozen bits.

For an $\left(N, K, A, u_A^C\right)$ polar code, A represents the index set of information bits u_A, while A^C denotes that of the frozen bits u_A^C. SC decoding of polar codes is predicated on sequentially processing u_i for $i = 0, 1, 2, \ldots, N-1$, under the assumption that all previous values of u_i have been correctly decoded. Consequently, the decoder functions by computing the likelihood parameters $L_N^{(i)}\left(y_0^{N-1}, \hat{u}_0^{i-1}\right) = \frac{W_N^{(i)}\left(y_0^{N-1}, \hat{u}_0^{i-1} \middle| 0\right)}{W_N^{(i)}\left(y_0^{N-1}, \hat{u}_0^{i-1} \middle| 1\right)}$, upon receiving the sequence $y = (y_0, y_1, \ldots, y_{N-1})$, which is a noise-corrupted version of x. Sequentially

for $i = 0, 1, 2, \ldots, N-1$, hard decisions on each u_i are made according to the rule $\hat{u}_i \triangleq \begin{cases} 0 & \text{if } u_i \text{ is a frozen bit} \\ 0 & \text{if } L_N^{(i)}\left(y_0^{N-1}, \hat{u}_0^{i-1}\right) \geq 1 \\ 1 & \text{otherwise} \end{cases}$.

2.2. **Neural channel decoding.** It is known that the deep neural network model can be abstracted as a function that maps the input $x_0 \in R^{N_0}$ to the output $y \in R^{N_L}$ using the formula $y = f(x_0; \theta)$. Here, θ represents the optimal parametric solution for the mapping between known inputs and expected outputs. The system framework and architecture of the DNN decoder are depicted in Figures 1 and 2, respectively. Specifically, the coded codeword x is mapped to the modulation sequence s through BPSK modulation, as described by the equation $s = -2x + 1$.

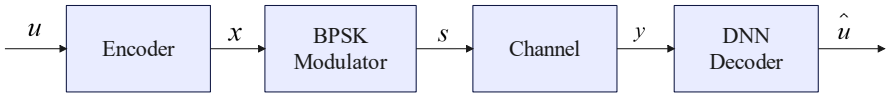


FIGURE 1. Framework of deep neural network decoder system

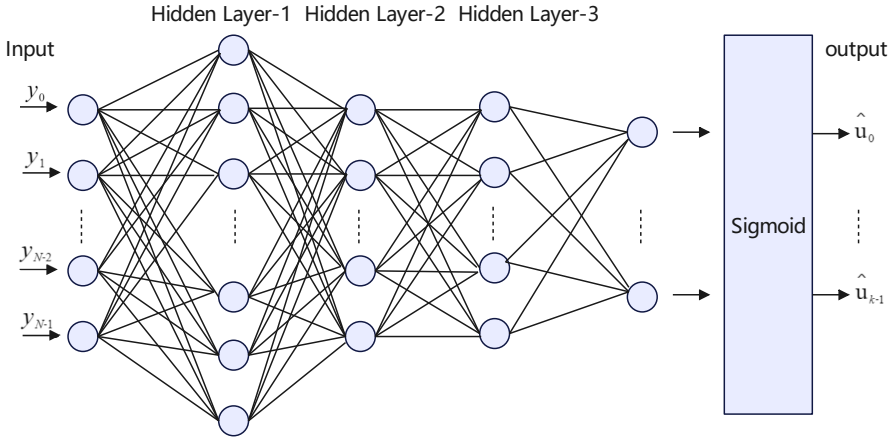


FIGURE 2. Architecture of deep neural network decoder

After the modulated sequence passes through the channel, noise interference m is added, resulting in the sequence y contaminated by noise. The receiving sequence y satisfies the equation $y = s + m$. During the training phase of the neural network, the goal is to minimize the loss function. Multiple training samples are used to adjust the neural network weights and biases, thereby obtaining the mapping function f. In the testing stage, the function f is used to estimate the new received symbols vector, a process known as one-hot coding. The deep neural network decoder incorporates a Rectified Linear Unit (ReLU) as the activation function in each hidden layer calculation unit for optimization. This function is defined as $f_{\mathrm{ReLU}}(x) = \max\{0, x\}$. The activation function selected for the output layer is the Sigmoid function, which readjusts the output to the range of $[0, 1]$. This function is chosen as $g_{sigmoid}(z) = \frac{1}{1+e^{-z}}$. To evaluate the decoding performance, the binary cross-entropy (BCE) loss function or the mean squared error (MSE) loss function is used to assess the neural network. These functions are defined as follows:

$$L_{BCE} = -\frac{1}{k}\sum_i \left[b_i \ln\left(\hat{b}_i\right) + (1 - b_i)\ln\left(1 - \hat{b}_i\right)\right] \tag{1}$$

$$L_{MSE} = \frac{1}{k} \sum_{i} \left(b_i - \hat{b}_i \right)^2 \tag{2}$$

where $b_i \in [0,1]$ and $\hat{b}_i \in [0,1]$ represent the ith transmission information bit and the ith estimated value, respectively. It is known that back propagation (BP) and stochastic gradient descent (SGD) form the core technology pair in deep learning model training. Specifically, back propagation calculates the gradients of the loss function with respect to the layer parameters using the chain rule, while SGD iteratively updates these parameters using the calculated gradients to minimize the loss function. Together, they ensure network convergence. Therefore, this model employs this combination to minimize the loss function until the network converges.

3. **Proposed Low-Precision SC Decoding Algorithm Based on DNN.** Normalized validation error (NVE) is used to evaluate the performance of a trained model when the training data is augmented with varying signal-to-noise ratio (SNR) noises. The expression for this is

$$NVE(\rho_t) = \frac{1}{S} \sum_{S=1}^{S} \frac{BER_{NND}(\rho_t, \rho_{v,s})}{BER_{MAP}(\rho_{v,s})} \tag{3}$$

Here, ρ_t represents the signal-to-noise ratio (SNR) of the specified training noise. The experimental SNR range chosen for this study is $\{-3, -2, -1, 0, 1, 2, 3, 4, 5, 6\}$. The variable S signifies the size of the verification data sets with a specified SNR for the verification data sets. We use $\rho_{v,s}$ to denote the verification set with an SNR of ρ_v. This index is utilized to assess the average bit error rate (BER) between the training data SNR and the maximum a posteriori probability decoding across various verification data SNRs.

Figure 3 illustrates the NVE value of the DNN decoder with a training hidden layer size of 128-64-32 and an epoch number of 2^{16}. Given that the E_b/N_0 distribution of the training data spans $\{-3, -2, -1, 0, 1, 2, 3, 4, 5, 6\}$, the NVE value initially decreases and subsequently rises, reaching its minimum at an SNR of 1. Consequently, the optimal SNR for the training data is determined to be 1, at which point the neural network decoder exhibits the best decoding error rate performance.

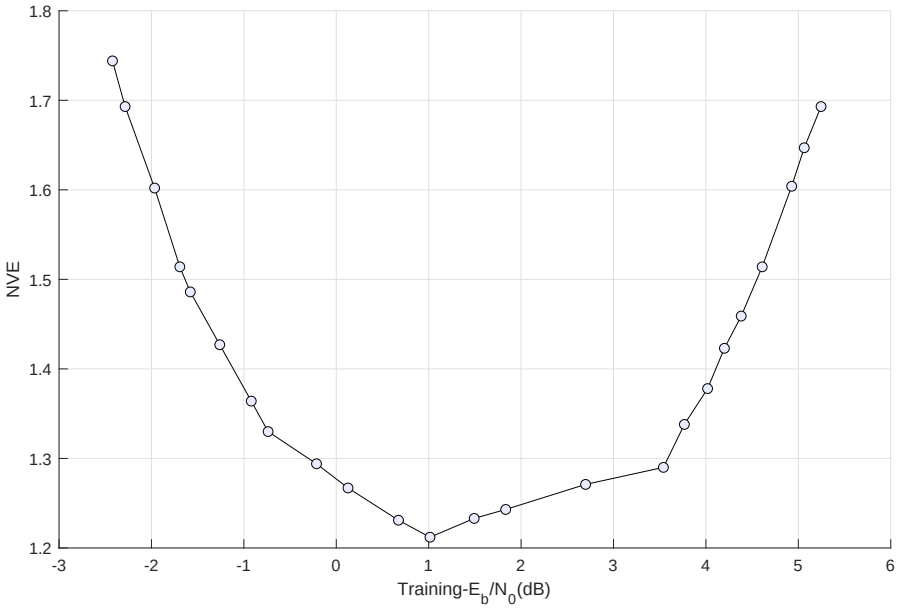


FIGURE 3. NVE under different SNR

3.1. **Optimization of activation function.** The Swish function is a novel activation function introduced by Google [17]. It is characterized by its unsaturated, smooth, and non-monotonic properties. The Swish function demonstrates exceptional performance across numerous datasets. The graphical representation of the Swish function on the positive half of the X-axis resembles that of the ReLU function. However, the Swish function's graph is smoother, which allows it to preserve the advantageous characteristics of the ReLU function on the positive half of the X-axis. Additionally, there are slight smooth undulations on the negative half of the axis, which effectively mitigate the issue of gradient vanishing caused by the constant value of 0. This contributes to enhancing the training efficiency and the error-correction capabilities of neural network decoders. A comparison of the graphical representations between the Swish function and the ReLU function is depicted in Figure 4, where the parameter value for the Swish function is set to $a = 1$. The formula for the Swish activation function is as follows:

$$f_{swish}(x) = x * sigmoid(ax) = \frac{x}{1 + e^{-ax}} \tag{4}$$

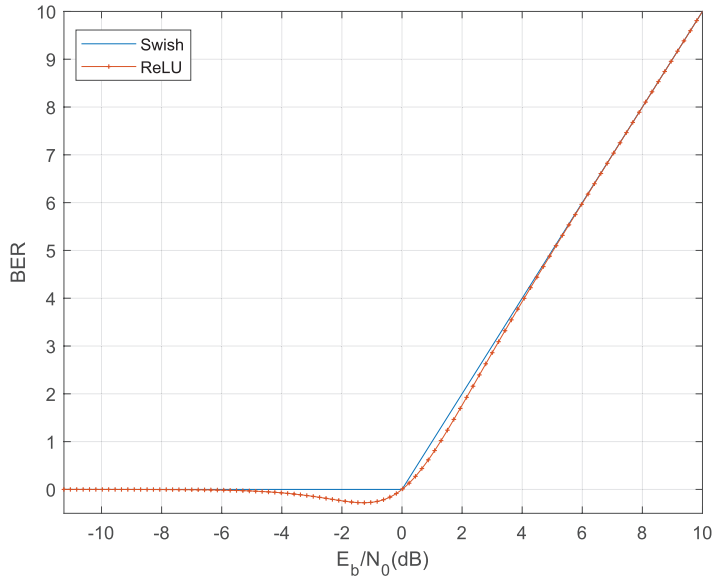


FIGURE 4. ReLU function image and Swish function image

As depicted in Figure 5, for the DNN decoder equipped with a $(16, 8)$ polar code, the BER performance between the ReLU activation function and the Swish activation function exhibits minimal difference at low signal-to-noise ratios. However, at medium to high signal-to-noise ratios, notably at 5 dB, the DNN decoder employing the Swish function demonstrates a substantial performance enhancement. Overall, the FER performance of the DNN decoder utilizing the Swish activation function surpasses that of the DNN decoder using the ReLU activation function.

3.2. **Optimization of loss function.** The traditional DNN decoder employs identical weights for all N-bit decoding outcomes when calculating the loss function. This means that each bit receives equal attention and is expected to predict the correct result uniformly. Consequently, when computing the derivative of back propagation through the loss function, each bit is treated indistinguishably. However, this approach is akin to fitting N random binary numbers and outputting a set of random numbers, which significantly increases the network's learning complexity.

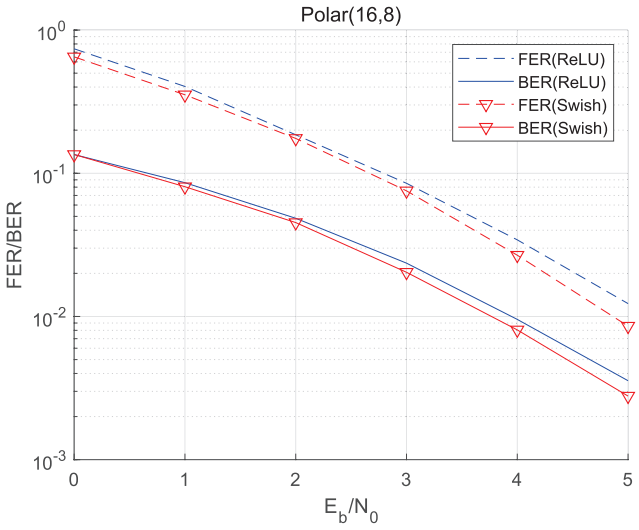


FIGURE 5. Decoding performance diagrams of two activation functions

Although neural networks can theoretically utilize any function, obtaining an optimal function is challenging due to the requirement for numerous parameters. Most importantly, even if any function can be used, there is no guarantee that the selected function can serve as a polar code decoder. Polar codes, however, differ from other unstructured codes due to their polarization properties, which result in a significant disparity in the importance of decoding outcomes. To better assist neural networks in learning the characteristics of a polar decoder, this paper proposes an improvement to the binary cross-entropy loss function by adopting the weighted binary cross-entropy (WBCE) loss function. Specifically, the loss function value corresponding to the estimated value of each bit during the decoding process is multiplied by a corresponding coefficient to achieve the effect of different weights for samples. Simulation results demonstrate that this approach enables the neural network to learn to become a more effective polar decoder and exhibits superior generalization capabilities.

The improved weighted loss function multiplies the value of the loss function corresponding to the decoding result of each bit by the weight of the respective bit. The weight value is determined by the BER performance of the corresponding bit. Therefore, bits with a higher BER have a more significant impact on the loss function, while those with a lower BER have a less impact, more accurately replicating the decoding characteristics of polar codes.

In AWGN channels, the bit error probability of a channel $P\left(W_N^{(i)}\right)$ can be obtained through Gaussian approximation, where the variable $W_N^{(i)}$ represents each polarized subchannel. We sort the bit error probabilities of all channels in ascending order and select the subscripts of the smallest k elements as the positions of the transmitted information bits. After obtaining the bit error probabilities of the channels, we can express the weight coefficient of the loss function corresponding to the ith bit as $w_i = 1 - P\left(W_N^{(i)}\right)$.

This implies that for bits with low theoretical error probabilities, the contribution of the loss function due to its decoding error should be greater than that for bits with high bit error probabilities. The corresponding improved binary weighted cross entropy loss function expression can be described as

$$L_w(w_i) = -\frac{1}{k}\sum_i \left[b_i \ln\left(\hat{b}_i\right) + (1-b_i)\ln\left(1-\hat{b}_i\right)\right] \tag{5}$$

For the DNN decoder of a $(16, 8)$ polar code, the BCE loss function and WBCE loss function are employed for network training, respectively. Figure 6 illustrates that there is virtually no difference in BER and FER performance when the SNR is in low regions. However, at high SNR levels, particularly between 4dB and 5dB, the performance of the DNN decoder utilizing the WBCE loss function is enhanced to a certain degree. Furthermore, Figure 7 demonstrates that during the training of the DNN decoder for a $(16, 8)$ polar code, the improved weighted binary cross-entropy loss function converges more rapidly than the standard binary cross-entropy loss function. This occurs because the weighted cross entropy loss function can more effectively balance the weights of various categories during the training of neural network decoders, thereby improving the model's generalization ability and classification effect, allowing the neural network to better learn the characteristics of polar codes.

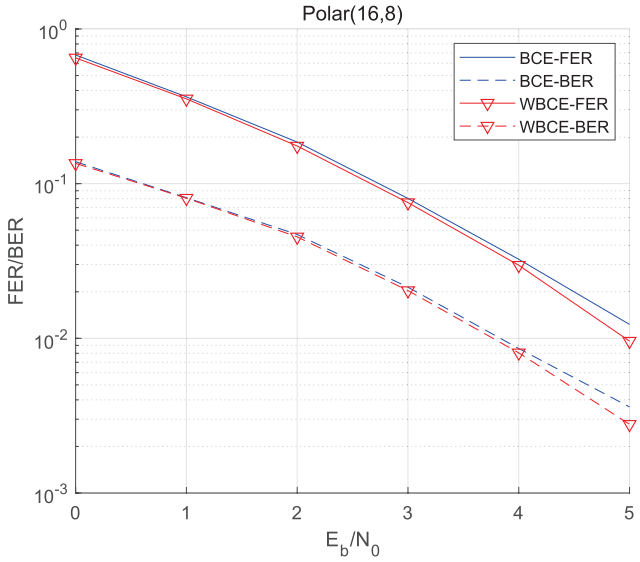


FIGURE 6. Decoding performance diagrams of two loss functions

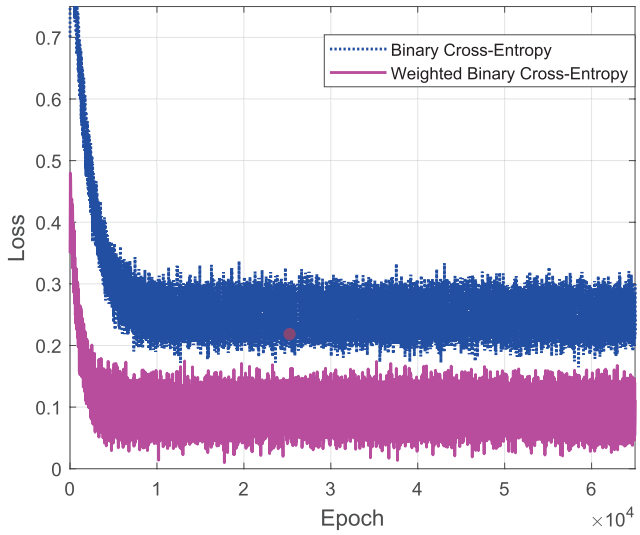


FIGURE 7. Two loss functions of DNN decoder

3.3. **Weight quantification of DNN decoder.** Given that it must store neural network weights and a variety of data types, memory usage leads to increased energy consumption. It is understood that the floating-point operations required result in higher memory resource and time overheads, which in turn limit the amount of data that can be processed in parallel. Inspired by [18], the proposed decoder reduces memory resource overhead by quantizing the weights to a limited number of bits. All arithmetic operations are conducted using an 8-bit quantization input and output in the Q8.4 fixed-point digital format. Consequently, weights can be stored in an 8-bit fixed-point data format, which employs $t-1$ bits to represent the decimal part and one bit for the sign, where t is the number of bits allocated for quantization. In the fixed-point data format, the bit width and resolution directly influence calculation accuracy. A wider bit width and higher resolution can enhance computational accuracy, but they also increase computational complexity and power consumption. Since the proposed scheme requires quantizing the neural network to enable int8 calculations, the number of bits t used for quantization is set to 8. Moreover, an int8 is an 8-bit signed integer, where one bit is used to represent the sign and the remaining 7 bits represent the value. Consequently, an int8 occupies 8 bits, or one byte of storage space, effectively reducing memory resource overhead. To mitigate the impact of quantization on BER performance post-training, quantization perceptual training is adopted, allowing quantization errors to be incorporated into the training loss. This enables the neural network to learn how to decode with limited accuracy. Since TensorFlow does not support local training with less than 8 bits, quantization errors are introduced into the neural network using a fake quantization function, and the quantization process is illustrated in Figure 8.

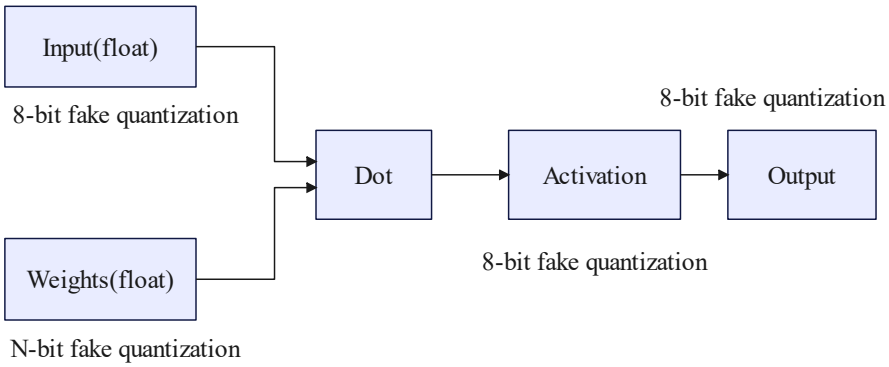


FIGURE 8. Weighted quantization data flow of DNN decoder

The fake quantization function is a deep learning tool that emulates the effects of quantization, supported by mainstream frameworks such as TensorFlow and PyTorch. During model training, it replicates the numerical truncation and rounding errors associated with low-precision quantization (e.g., 8-bit integers), allowing model parameters and activations to adapt to precision loss induced by quantization in advance. This, in turn, enhances deployment efficiency.

3.4. **Low-precision DNN-SC decoding algorithm.** The complexity of employing neural networks directly as decoders increases exponentially with the growth of information bits. Inspired by the partitioning concept of the neural network-aided SC decoder (NNSC) proposed in [11], polar codes are segmented using neural network decoding, assisted by the traditional SC decoder. The selection of information bits for polar codes is dictated by the channel structure. Typically, the number of segments is chosen based on the

specific implementation of the SC algorithm and the characteristics of the hardware architecture. Generally, the number of segments should be a power of two to facilitate recursive computation. Consider an $(8, 4)$ polar code with the information bit index set $\{3, 5, 6, 7\}$ and a $(16, 8)$ polar code with the information bit index set $\{9, 10, 12, 7, 11, 13, 14, 15\}$ as examples for dividing the long code. If the $(16, 8)$ polar code is divided into two 8-length code blocks, the distribution of information bits can be observed in Table 1.

TABLE 1. Information bit distribution of polar codes partitioning

Partitioning	Information bit	Code rate
[0-7]	{7}	0.125
[8-15]	{9,10,11,12,13,14,15}	0.875

As indicated in Table 1, the $(16, 8)$ polar code is split into two 8-length codes with a code rate of 0.125 for the first block and 0.875 for the second block. These codes are trained through partitioning and subsequent splicing. Given the structural properties of polar codes, partitioning and splicing necessitate the use of the traditional SC decoding architecture. The SC decoder involves the splicing of computing units, where each unit, consisting of 2 inputs and 2 outputs, encompasses two functions: f and g. The f function can be computed solely based on the two y inputs, whereas the g function requires not only the two y inputs but also the output of the f function. Taking the code block of length 8 as an example, the model and flow chart of the DNN-SC decoder are depicted in Figures 9 and 10. The computations for functions f and g are as follows:

$$\begin{aligned} &LLR_N^{(2i-1)}\left(y_0^{N-1}, \hat{u}_0^{2i-2}\right) \\ &= 2\tanh^{-1}\left(\tanh\left(\frac{LLR_{N/2}^{(i)}\left(y_0^{N/2-1}, \hat{u}_{0,o}^{2i-2} \oplus \hat{u}_{0,e}^{2i-2}\right)}{2}\right) \cdot \tanh\left(\frac{LLR_{N/2}^{(i)}\left(y_{N/2}^{N-1}, \hat{u}_{0,e}^{2i-2}\right)}{2}\right)\right) \end{aligned} \tag{6}$$

$$\begin{aligned} &LLR_N^{(2i)}\left(y_0^{N-1}, \hat{u}_0^{2i-1}\right) \\ &= (-1)^{\hat{u}_{2i-2}} LLR_{N/2}^{(i)}\left(y_0^{N/2-1}, \hat{u}_{0,o}^{2i-2} \oplus \hat{u}_{0,e}^{2i-2}\right) + LLR_{N/2}^{(i)}\left(y_{N/2}^{N-1}, \hat{u}_{0,e}^{2i-2}\right) \end{aligned} \tag{7}$$

For an (N, K) polar code, the encoded sequence x derived from u is used for training, corresponding to an N-bit input and output, which is divided into two parts for separate training. The training set for the first DNN decoding partition module is the result of the f calculation for each y, and the training label is the first K bits of x. As for the second DNN decoding partition module, its training set is the result of the g calculation for y.

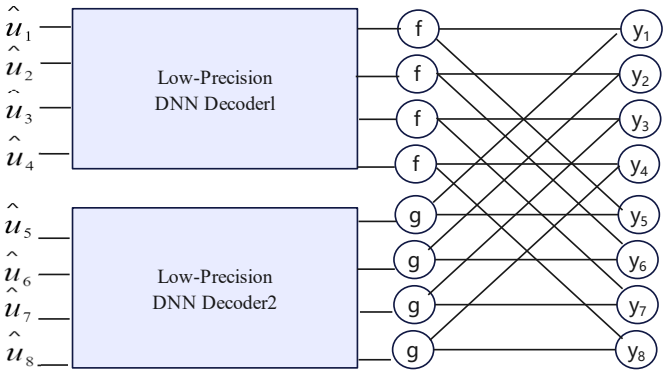


FIGURE 9. Decoding model of low-precision DNN-SC algorithm

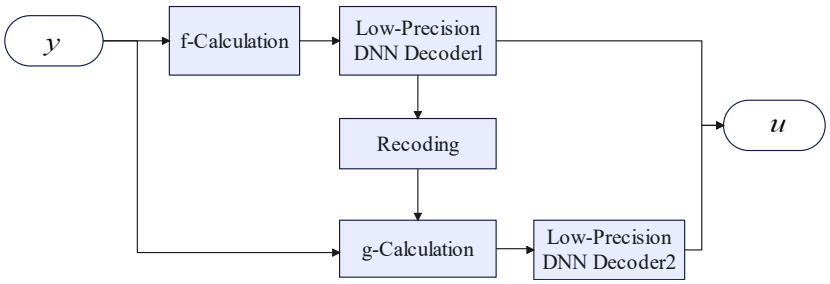


FIGURE 10. Decoding flow of low-precision DNN-SC algorithm

After the first DNN decoding partition module is recorded, its training label becomes the last K bits of each codeword x. This is because the number of information bits and the number of neurons in the two partition decoding modules are different, specifically 128 and 256, respectively. Finally, the decoding result can be obtained by combining the outputs from the two DNN decoding partition modules.

4. **Simulation Results and Analysis.** We adopted the Kreas DNN framework. The neural network decoder was designed and implemented using the TensorFlow backend, employing the Adam gradient descent optimization algorithm with a learning rate of 0.001 for training. To obtain suitable training and validation datasets, we initially generate 1 million random information sequences and encode them into polar codewords for AWGN channels. White Gaussian noise is then imposed on the encoded data, and subsequently, the correctly decoded codewords are compiled. The LLR values, frozen bit patterns, and correct codewords are compiled within the SC decoder. These random codewords encompass sub-block decoding outcomes under various frozen bit patterns and different E_b/N_0 ratios. Thereafter, 90% of the random codewords are designated as training sets, while the remaining 10% serve as validation sets. The number of training epochs and batch size are set to 2^{16} and 256, respectively.

4.1. **Decoding performance analysis.** We compare the BER performances of the proposed decoder with those of the DNN [10], NNSC [11], SC, and BP decoders for $(16, 8)$ and $(32, 16)$ polar codes, respectively. The BER curves are shown in Figures 11 and 12.

Figures 11 and 12 illustrate that the BER performance diminishes when employing a DNN decoder directly. The BER performance of a DNN decoder with a codeblock

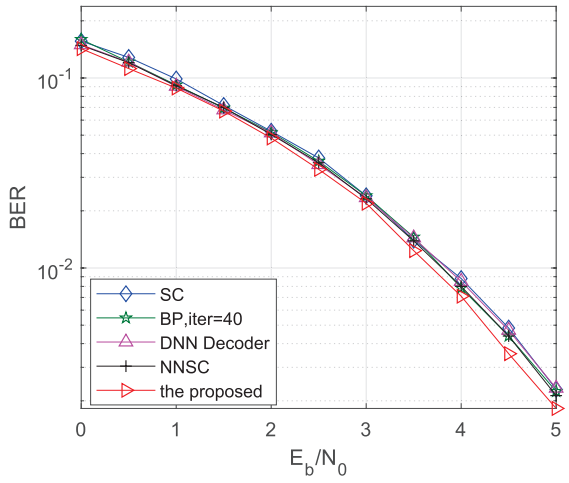


FIGURE 11. BER performance with a $(16, 8)$ polar code under different decoders

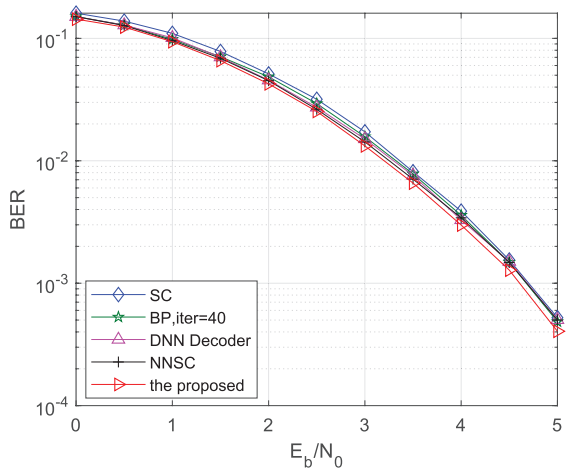


FIGURE 12. BER performance with a $(32, 16)$ polar code under different decoders

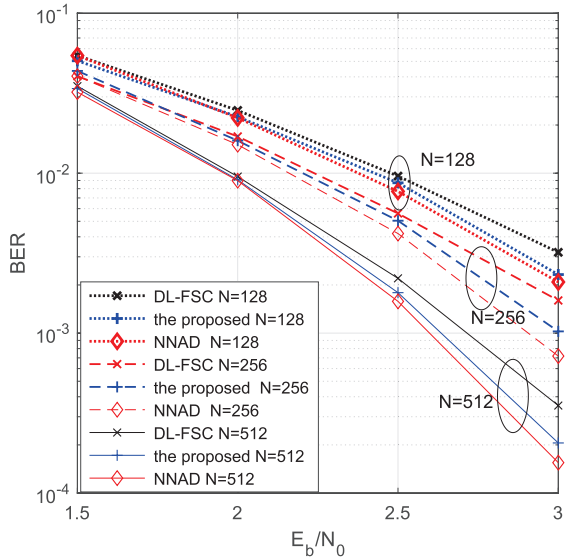


FIGURE 13. BER performance comparison of polar codes with different code block lengths ($N = 128, 256, 512$) under various decoders

size of 16 is comparable to that of a BP decoder with 40 iterations. In contrast, the BER performance of the SC decoder is marginally superior to that of both the neural network decoder and the BP decoder. Nevertheless, the decoding performance of the low-precision DNN-SC decoder proposed in this paper surpasses that of the SC decoder and the NNSC decoder. Notably, at high SNR regions, the performance improvement is substantial. Compared to the NNSC decoder, the proposed decoder achieves a performance gain of approximately 0.20 dB at a BER of 2×10^{-2}, and a performance gain of 0.16 dB at a BER of 5×10^{-3}.

Additionally, we also investigate the decoding performance of longer code blocks of polar codes under the proposed scheme. Figure 13 demonstrates that the BER performance comparsion of the proposed low-precision DNN-SC decoder with NNAD [19] and DL-FSC [20] decoders under the blocklengths 128, 256 and 512. Specifically, DL-FSC is a deep-learning-aided fast SC (DL-FSC) decoding algorithm which partitions SC decoder into multiple sub-DL decoders and connects them through the SC decoder. Although the decoder can support polar codes with any code rate under a certain code length to improve the decoding rate, it needs consume more logical resources. NNAD is also a neural

network-assisted decoding scheme (NNAD) that utilizes neural networks to decode special and key bit-based subcodes. The above decoding schemes are both based floating-point arithmetic. From the simulation results, we can see that the proposed scheme performs better than DL-FSC and is slightly suboptimal to NNAD but with a lower complexity.

4.2. **Analysis of decoding latency and weight storage.** For an N-length polar code and the number of BP iterations I, the latency of BP decoding can be expressed in time steps as $T_{BP} = 2I\log_2 N$. From [1] and [11], it is known that the latency of SC decoder and NNSC decoder with stage index S can be expressed as $T_{SC} = 2N - 2$ and $T_{NNSC} = \frac{N}{2^S}(T+1) + 2\frac{N}{2^S} - 2$, respectively. The decoding latency required by the proposed low-precision DNN-SC decoder is the same as that of the NNSC decoder. The available time steps can be expressed as $T_{proposed} = \frac{N}{N_0}\left(\log\frac{N}{N_0} + T + 1\right)$ with the size of the code partition block N_0 and the number of hidden layers T in the network. Neural network decoders are trained and coupled with the SC decoding algorithm, and their decoding performance and latency are related to the number of subblocks in the proposed scheme. In general, using more subblocks can improve decoding performance because it provides more decoding information and greater error correction. However, employing more subblocks also increases the complexity and computation of the neural network, resulting in higher latency and power consumption. Therefore, selecting the appropriate number of subblocks is a trade-off process that can be evaluated from simulation results. For a $(32, 16)$ polar code, the latency of various decoders is shown in Table 2.

Table 2. Decoding latency (time steps)

Algorithm	BP	SC	NSC	Proposed
Latency	400	62	10	9

From Table 2, we can see that the proposed decoder requires 10% fewer time steps and 85.4% fewer time steps than the SC decoder when compared with the NNSC decoder at a stage index S of 4. Table 3 illustrates that the proposed decoder, employing weight quantization, utilizes an 8-bit quantization in the Q8.4 fixed-point digital format for training a $(16, 8)$ neural network decoder. This approach reduces the weight of the neural network decoder and diminishes the high memory requirements associated with floating-point decoding. From Table 4, it is evident that the proposed scheme's calculations are faster and less complex than those using DL-FSC or NNAD decoders,

Table 3. Weights of different quantization schemes

Hide layer size	$(512, 256, 128)$	$(128, 64, 32)$	$(128, 64, 32)$
Weight type	float	float	int8
Size (bytes)	716800(100%)	73728(10.3%)	12888(1.8%)

Table 4. Reduction gain in computational complexity of different decoding schemes relative to conventional SC decoding algorithm

Decoding scheme	Reduction gain per block length (bits)		
	128	256	512
DL-FSC	56.19%	57.23%	58.42%
NNAD	34.21%	36.07%	37.85%
The proposed scheme	74.06%	75.51%	76.61%

which are both deep learning-aided decoding schemes employing high-precision floating-point arithmetic. The reduction in relative computational complexity is expressed as $((C_{sc}(N) - C_{improved}(N))/C_{sc}(N))$, where $C_{improved}(N)$ represents the computational time of the DL-FSC, NNAD, or the proposed scheme, respectively. Table 4 indicates that the proposed scheme reduces computational complexity by approximately 75% compared to the original SC decoding algorithm, which is higher than the 58% and 35% reductions achieved by the DL-FSC and NNAD schemes, respectively. This clearly demonstrates that the proposed scheme can significantly decrease the computational complexity in polar codes decoding.

5. **Conclusions.** Combining the concept of partitioning, the proposed scheme initially segments a polar code into multiple blocks and decodes them using neural networks within the SC decoding subtree. Neural network decoders are then employed to partially substitute the traditional serial cancellation decoding in the subsequent layers. The proposed decoder is implemented by quantizing the weights of the deep neural network decoder to a limited number of bits. Subsequently, 8-bit quantized input and output in Q8.4 fixed-point digital format are utilized for all arithmetic operations, enabling int8 computation. Additionally, with quantization-aware training, quantization error is incorporated into the training loss, allowing the neural network to effectively learn how to decode with limited precision. Simulation results indicate that the proposed algorithm not only enhances BER performance to a certain extent but also achieves lower decoding latency while reducing memory overhead.

However, quantization errors are inevitably introduced during the quantization of DNN parameters, which may degrade the performance of SC decoding, particularly in high-precision application scenarios. Consequently, we plan to explore an adaptive strategy that can dynamically balance the trade-off between accuracy and performance in future work.

Acknowledgment. This work has received partial support from the Key Research Projects of Higher Education Institutions in Henan Province, under grant number 25A413012; in part from the Science and Technology Plan Project of Weiyang District in Shaanxi Province, under Grant 202424; in part from the Key Research and Development General Project of Shaanxi Provincial Science and Technology Program, under Grant 2025CY-YBXM-006; and in part from the Scientific and Technological Project of Henan Province, under grant number 252102211010. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

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] R. Joshi and J. Sattar, One-shot gesture recognition for underwater diver-to-robot communication, *arXiv Preprint*, arXiv: 2503.00676, 2025.

[3] I. Tal and A. Vardy, List decoding of polar codes, *IEEE Transactions on Information Theory*, vol.61, no.5, pp.2213-2226, 2011.

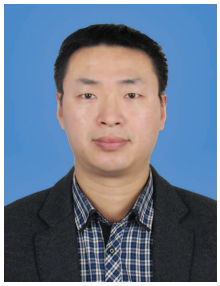
[4] K. Niu and K. Chen, CRC-aided decoding of polar codes, *IEEE Communications Letters*, vol.16, no.10, pp.1668-1671, 2012.

[5] S. Miao, C. Kestel, L. Johannsen, M. Geiselhart, L. Schmalen, A. Balatsoukas-Stimming, G. Liva, N. Wehn and S. T. Brink, Trends in channel coding for 6G, *Proceedings of the IEEE*, 2024.

[6] E. Nachmani, Y. Be'ery and D. Burshtein, Learning to decode linear codes using deep learning, *Proc. of the 2016 54th Annual Allerton Conference on Communication, Control, and Computing, Monticello*, pp.341-346, 2016.

[7] T. Gruber, S. Cammerer, J. Hoydis and S. T. Brink, On deep learning-based channel decoding, *Proc. of the 51st Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, pp.1-6, 2017.
[8] E. Nachmani and L. Lugosch, Deep learning methods for improved decoding of linear codes, *IEEE Journal of Selected Topics in Signal Processing*, vol.12, no.1, pp.119-131, 2018.
[9] T. Matsumine and H. Ochiai, Recent advances in deep learning for channel coding: A survey, *IEEE Open Journal of the Communications Society*, vol.5, pp.6443-6481, DOI: 10.1109/OJCOMS.2024.3472094, 2024.
[10] S. Cammerer, T. Gruber, J. Hoydis and S. T. Brink, Scaling deep learning-based decoding of polar codes via partitioning, *Proc. of the IEEE Global Communications Conference*, Singapore, pp.1-6, 2017.
[11] N. Doan, S. A. Hashemi and W. J. Gross, Neural successive cancellation decoding of polar codes, *Proc. of the IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, pp.1-5, 2018.
[12] Y. Lee, U. Lee, H. H. Fisseha and M. H. Sunwoo, Deep learning aided BP-flip decoding of polar codes, *Proc. of IEEE 4th Int. Conf. Artificial Intell. Circuits Syst.*, Incheon, Korea, pp.114-117, 2022.
[13] W. Li, Q. Tian, Y. Zhang, T. Feng, Z. Li, Q. Zhang and Y. Wang, A rate-compatible punctured Polar code decoding scheme based on deep learning, *Proc. of Int. Conf. Optical Commun. Netw.*, Shenzhen, China, pp.1-3, 2022.
[14] S. Cammerer, J. Hoydis, F. A. Aoudia and A. Keller, Graph neural networks for channel decoding, *Proc. of IEEE GC Wkshps*, pp.486-491, 2022.
[15] Y. Shang, Z. Zhang and Z. Yang, LSTM-based path selection for successive cancellation list decoding for short polar codes, *Proc. of IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp.1-5, 2023.
[16] J. Li et al., Deep learning-assisted adaptive dynamic-SCLF decoding of polar codes, *IEEE Trans. Cognit. Commun. Netw.*, vol.10, no.3, pp.836-851, 2024.
[17] P. Ramachandran, B. Zoph and Q. V. Le, Searching for activation functions, *arXiv Preprint*, arXiv: 1710.05941, 2017.
[18] Wodiany and A. Pop, Low-precision neural network decoding of polar codes, *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Cannes, France, pp.1-5, 2019.
[19] H. Liu, L. Zhang, W. Yan and Q. Ling, Neural-network-assisted polar code decoding schemes, *Applied Sciences*, DOI: 10.3390/app122412700, 2022.
[20] 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, 2024.

Author Biography



Jie Shen received the B.S. degree and the M.S. degree from Zhengzhou University, China, in 2004 and 2007, respectively. He is currently a teacher in the School of Mechanical Engineering, North China University of Water Resources and Electric Power, China. His current research interests include signal acquisition and processing, embedded system and its application.

Chang Yun received the B.S. degree in Computer Science and Technology from Chongqing Jiaotong University, China, in 2020. During 2020-2023, he stayed in Xi'an Technological University, China, to study for a M.S. degree. Recently, he has worked in Hangzhou on communication software design. His current research includes IoT, channel coding, robot communication and its applications.

Wenfan Wang received the B.S. degree from Henan Normal University, China, in 2004, and the M.S. degree from Zhengzhou University, China, in 2007. She is currently an associate professor in School of Information Engineering, Henan Vocational College of Water Conservancy and Environment, China. Her current research interests include artificial intelligence, big data and computer applications.

Guiping Li received the B.S. degree from Zhengzhou University, China, in 2000, the M.S. degree from Xi'an University of Science and Technology, China, in 2006, and the Ph.D. degree from Xidian University, China, in 2016. She is currently an associate professor in School of Computer Science and Technology, Xi'an Technological University, China. Her current research interests include communication information theory, channel coding theory and their applications.