entropy

MDPI

Article

DLSTM-Based Successive Cancellation Flipping Decoder for Short Polar Codes

Jianming Cui [1], Wenxiu Kong [1], Xiaojun Zhang [1,2,*], Da Chen [1] and Qingtian Zeng [1]

[1] School of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China; cuijm399@163.com (J.C.); kwx1836092326@163.com (W.K.); chenda@sdust.edu.cn (D.C.); qtzeng@sdust.edu.cn (Q.Z.)
[2] State Key Laboratory of High-End Server and Storage Technology, Jinan 250101, China
* Correspondence: buttern@163.com

Abstract: Polar code has been adopted as the control channel coding scheme for the fifth generation (5G), and the performance of short polar codes is receiving intensive attention. The successive cancellation flipping (SC flipping) algorithm suffers a significant performance loss in short block lengths. To address this issue, we propose a double long short-term memory (DLSTM) neural network to locate the first error bit. To enhance the prediction accuracy of the DLSTM network, all frozen bits are clipped in the output layer. Then, Gaussian approximation is applied to measure the channel reliability and rank the flipping set to choose the least reliable position for multi-bit flipping. To be robust under different codewords, padding and masking strategies aid the network architecture to be compatible with multiple block lengths. Numerical results indicate that the error-correction performance of the proposed algorithm is competitive with that of the CA-SCL algorithm. It has better performance than the machine learning-based multi-bit flipping SC (ML-MSCF) decoder and the dynamic SC flipping (DSCF) decoder for short polar codes.

Keywords: polar codes; 5G; SC decoding; bit flipping; LSTM; robustness

check for updates

Citation: Cui, J.; Kong, W.; Zhang, X.; Chen, D.; Zeng, Q. DLSTM-Based Successive Cancellation Flipping Decoder for Short Polar Codes. *Entropy* **2021**, *23*, 863. https://doi.org/10.3390/e23070863

Academic Editor: Raúl Alcaraz

Received: 20 May 2021
Accepted: 30 June 2021
Published: 6 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Polar codes have been proven to achieve channel capacity [1], thus were selected as the coding standard for the control channels of enhanced mobile broadband (eMBB) scenarios in 2016 [2]. Normally, the block length of the control channel is short or moderate, but Arıkan [3] proved that channel polarization occurs only when the number of channels gradually becomes larger. Therefore, the performance of polar codes is not satisfactory at short or moderate block length due to insufficient channel polarization.

The successive cancellation (SC) decoding [3] has a regular-recursive structure, but the performance is uncompetitive under the finite block length. In order to improve error-correction performance, the succession cancellation list (SCL) and the successive cancellation flipping (SC flipping) algorithms are introduced. Among them, Niu et al. proposed a cyclic redundancy check (CRC) aided SCL algorithm (CA-SCL) [4], which bridges the gap between SC and maximum-likelihood algorithms. The CA-SCL decoding improves performance compared to the conventional SCL decoding [5], but both algorithms come at the cost of high computational complexity. The SC flipping algorithm [6] can achieve a similar computational complexity as the SC decoding, which competes with the CA-SCL decoding. The flipping algorithm constructs the flipping set by the absolute value of log-likelihood ratio (LLR) to locate the first error bit, while there are additional error bits in subsequences. On this basis, a multi-bit flipping algorithm [7–9] was proposed to correct multiple error bits. In [8], Zhang et al. defined a set of flipping positions as the critical set (CS). The LLR value is used to determine the flipping bit, which requires sorting the LLR sequence.

Since deep learning was adopted to locate the error bit more accurately than traditional methods [6–9], many scholars have applied the neural network to decoders [10–12]. In [13,14], a decoder based on the convolutional neural network (CNN) is proposed to lower the bit error rate (BER). Chun-Hsiang Chen et al. applied the long short-term memory (LSTM) network to the decoder [15]. The network can efficiently extract features from the decoding and improve the accuracy of error bit prediction, but the complexity increases with the list size. In order to obtain a better flipping strategy, both the LSTM and the reinforcement learning (RL) are applied to the decoder. In [16], the LSTM network predicts the first error bit and the RL identifies other error bits or undoes the previous error flipping. Similarly, a machine learning-based multi-bit flipping SC decoding scheme (ML-MSCF) was proposed by Author17 [17], which uses both the LSTM network and the Q-Learning algorithm to achieve multi-bit flipping. However, it needs a well-trained Q-table to select the flipping bit.

The accuracy of the flipping bit greatly affects the performance of the SC flipping decoder. However, the current SC flipping works do not perform well with short codewords. The LSTM networks have achieved excellent performance in many fields [18,19]. Thus, we propose a double long short-term memory (DLSTM) network to predict the error bits. The main contributions are as follows:

- A DLSTM-based SC flipping decoder is proposed. In addition, all frozen bits are clipping in the output layer to enhance accuracy. We construct the flipping set of the first error bit according to the probability of the network output in descending order, and attempt one-bit flipping.
- By exploring the reliability of bit channel, we propose a novel multi-bit flipping scheme for sequences that reach the maximum number of one-bit flipping and still fail the CRC detector. The candidate bits are sorted in ascending order in terms of the reliability, and the unreliable bits are selected in priority to multi-bit flipping.
- In order to make the proposed algorithm robust, we design the DLSTM network architecture to be compatible with multiple block lengths. We adopt a padding strategy to maintain data integrity, so that the training is not limited by the block length. A masking method is taken to skip the timestep and eliminate the effect of padded invalid data. Simulation results show that the proposed decoding scheme has better error-correction performance than the ML-MSCF decoding and DSCF decoding for short block lengths. It can approach the performance of CA-SCL ($L = 8$).

The remainder of the paper is organized as follows. Section 2 introduces polar codes, SC flipping decoder, and LSTM network. The proposed decoding scheme is presented in Section 3. Section 4 analyzes the decoding performance. Conclusions are drawn in Section 5.

2. Preliminary

In this paper, we use the calligraphic characters, such as $\mathcal{X}$, to denote sets. We write x, $\mathbf{x}$, and $\mathbf{X}$ to denote a scalar, a vector, and a matrix, respectively. In this section, we first review the basic of polar codes. Then, we briefly introduce the SC flipping decoder and LSTM network.

2.1. Polar Codes

A polar code of length N with K information bits can be represented by $P(N, K)$. The code rate is $R = K/N, 0 < K < N$. The K most reliable channels are selected to transmit the information bits and the remaining $N - K$ channels transmit the frozen bits. The set of information bit indices and frozen bit indices are denoted by $\mathcal{A}$ and its complement $\mathcal{A}^{\mathrm{C}}$. The sequence $\boldsymbol{u}_1^N = \{u_1, u_2, \ldots, u_N\}$ is obtained by mixing the information bits $u_{\mathcal{A}}$ and the frozen bits $u_{\mathcal{A}^{\mathrm{C}}}$, and a polar codeword $\boldsymbol{x}_1^N$ is generated by generator matrix $\mathbf{G}_N$:

$$\boldsymbol{x}_1^N = \boldsymbol{u}_1^N \mathbf{G}_N, \tag{1}$$

where $x_1^N = \{x_1, x_2, \ldots, x_N\}$, $\mathbf{G}_N = \mathbf{B}_N\mathbf{F}^{\otimes n}$, $n = \log_2 N$, $\mathbf{B}_N$ is a $N \times N$ bit-reversal permutation matrix, and $\mathbf{F}^{\otimes n}$ denotes the nth Kronecker power of $\mathbf{F}$, with

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} . \tag{2}$$

The butterfly-based decoding structure is shown in Figure 1. The SC decoding starts by computing an estimate value $\hat{u}_1$ of u_1 based only on the received value y_1^N. Then, y_1^N and $\hat{u}_{i-1}$ are used to estimate $\hat{u}_i$, $i = 2, \ldots, N$. A binary input memoryless channel W_n generates N sub-channels by channel splitting, defining $W_n^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|u_i\right)$, $i = 1, \ldots, N$. Let the LLR of $W_n^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|u_i\right)$ be defined as

$$L_n^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|u_i\right) \triangleq \log\left(\frac{W_n^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|u_i = 0\right)}{W_n^{(i)}\left(y_1^N, \hat{u}_1^{i-1}|u_i = 1\right)}\right). \tag{3}$$

The decision of LLR can be calculated by the f and g operations in Figure 1. The f and g operations are:

$$f(L_1, L_2) = \text{sign}(L_1)\text{sign}(L_2)\min(|L_1|, |L_2|) \ , \tag{4}$$

$$g(L_1, L_2, u) = (-1)^u L_1 + L_2 \ , \tag{5}$$

where $L_1, L_2 \in \mathbb{R}$ are the two inputs for the f and g operations and $u \in \{0, 1\}$ is the partial sum for g operation. The decision of LLR is expressed as

$$\begin{aligned} L_N^{(2i-1)}\left(y_1^N, \hat{u}_1^{2i-2}\right) = f\Big(&L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}\right), \\ &L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}\right)\Big), \end{aligned} \tag{6}$$

$$\begin{aligned} L_N^{(2i)}\left(y_1^N, \hat{u}_1^{2i-1}\right) = g\Big(&L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}\right), \\ &L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}\right), \hat{u}_{2i-1}\Big) . \end{aligned} \tag{7}$$

Finally, we obtain $\hat{u}_i$ as

$$\hat{u}_i = \begin{cases} 0, & L_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1}\right) \geq 0 \ and \ i \in \mathcal{A} \\ 1, & L_N^{(i)}\left(y_1^N, \hat{u}_1^{i-1}\right) < 0 \ and \ i \in \mathcal{A} \\ 0, & i \in \mathcal{A}^{\mathrm{C}} \end{cases} . \tag{8}$$

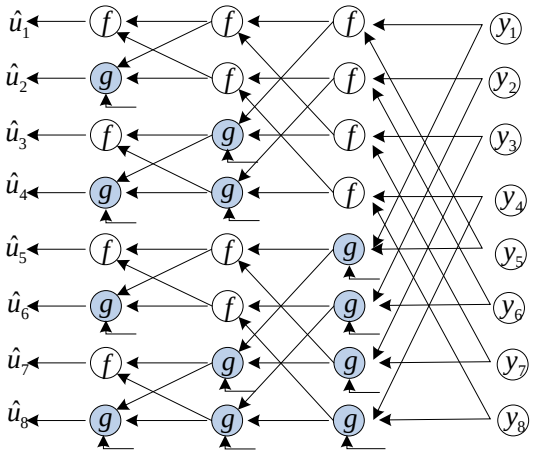


Figure 1. A butterfly-based decoder with $N = 8$.

2.2. SC Flipping Decoder

The SC flipping decoder initiates a SC decoder to generate the estimated codeword $\hat{u}_1^N$. If $\hat{u}_1^N$ passes the CRC detector, the decoding is successful. Otherwise, the flipping decoder is given T attempts to flip the first error bit. Algorithm 1 presents the original SC flipping algorithm, and the flipping set is denoted as $\mathcal{V}$. There are two reasons for decoding failures: channel noise and error propagation. The SC flipping algorithm improves error-correction performance and restrains error propagation. However, the decoder can only eliminate the error propagation caused by the first error bit, but there may be other error bits caused by noise or follow error bits in the subsequence. The information bits $\{9, 13\}$ in Figure 2, for example, are error bits caused by channel noise. The SC flipping decoder tries to corrects the first error bit $\{9\}$ caused by channel noise, and the error propagation phenomenon of $\{10, 11\}$ disappears, lowering the BER. However, $\{13\}$ is still an error and the block error rate (BLER) is not lower. Therefore, we consider flipping the second error bit $\{13\}$ to lower the BLER and BER in this paper.

Algorithm 1 The SC flipping algorithm.

Input: y_1^N, $\mathcal{A}$, T
Output: $\hat{u}_1^N$

$\{\hat{u}_1^N, \text{LLR}\} \leftarrow \text{SC}\ (y_1^N)$
if $\hat{u}_1^N$ fail CRC and $T > 1$ **then**
  $\mathcal{V} \leftarrow i \in \mathcal{A}$ of T smallest $\mid \text{LLR} \mid$
  for $j \leftarrow 1$ to T **do**
    $\hat{u}_1^N \leftarrow \text{SC}\ (y_1^N, \mathcal{V}(j))$
    if $\hat{u}_1^N$ pass CRC **then**
      break
    end if
  end for
end if
return $\hat{u}_1^N$

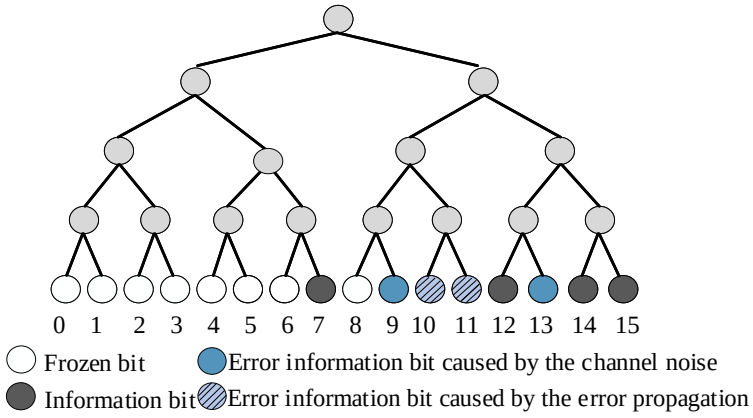


Figure 2. The SC decoding tree for a $P(16, 8)$.

2.3. LSTM Network

Compared with the traditional Recurrent Neural Network (RNN), the LSTM network is not only capable of handling long-term dependencies but also suitable for sequence-based tasks [20]. It is well known that the SC decoding generates the LLR sequences, and the LSTM networks can effectively extract features and build relationships between sequences. The structure of LSTM is shown in Figure 3, which consists of one state and three gates, namely cell state, forget gate, input gate, and output gate. The cell state $\mathbf{c}_t$ mainly saves the long-term state. The forget gate $\mathbf{f}_t$ determines the proportion of the previous cell state $\mathbf{c}_{t-1}$ retained to the current state $\mathbf{c}_t$. The input gate $\mathbf{i}_t$ decides the proportion of the input

$\mathbf{x}_t$ of the network saved to the cell state $\mathbf{c}_t$. The output gate $\mathbf{o}_t$ controls what message is output. The control of the three gates allows the LSTM network to remember more relevant messages for a long time, which is a great advantage when dealing with the LLR sequences. The gate is a fully connected layer, which can be expressed as:

$$\text{gate}(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{9}$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{x}$ represents the input vector, $\mathbf{b}$ is the bias vector, and σ is the sigmoid function:

$$\sigma(z) = \frac{1}{1+e^{-z}} \ . \tag{10}$$

The output $\mathbf{h}_t$ of the LSTM is defined by the output gate $\mathbf{o}_t$ and the cell state $\mathbf{c}_t$. The cell state $\mathbf{c}_t$ is determined by the forgot gate $\mathbf{f}_t$, the input gate $\mathbf{i}_t$, the cell state $\mathbf{c}_{t-1}$ at time $t-1$, and the input cell state $\mathbf{c'}_t$. In addition, the relationship at time t and $t-1$ is described as follows:

$$\begin{cases} \mathbf{f}_t = \sigma\left(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right) \\ \mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \\ \mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \\ \mathbf{c'}_t = \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \\ \mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{c}'_t \\ \mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \end{cases}, \tag{11}$$

where tanh is the activation function defined as:

$$\tanh(m) = \frac{e^{m}-e^{-m}}{e^{m}+e^{-m}} \ . \tag{12}$$

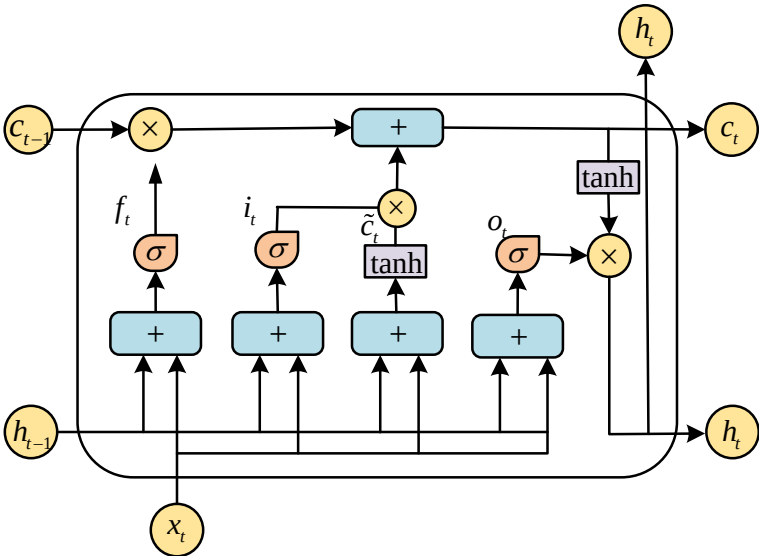


Figure 3. An LSTM basic unit.

3. The Proposed SC Flipping Algorithm

In Figure 4, we depict the frequency of errors caused by channel noise for $P(64, 32)$ at different E_b/N_0. It can be observed that the probability of one error bit occurring in the SC decoding is about 90% at high E_b/N_0, and there are still two or three error bits at low E_b/N_0. Therefore, in order to further improve the performance of the SC decoding, we also consider multi-bit flipping. In this section, we analyze the effect of error propagation. Then, we describe the DLSTM network structure and training process. Finally, we present the proposed DLSTM-based SC flipping algorithm and a mechanism to improve the robustness of the algorithm.

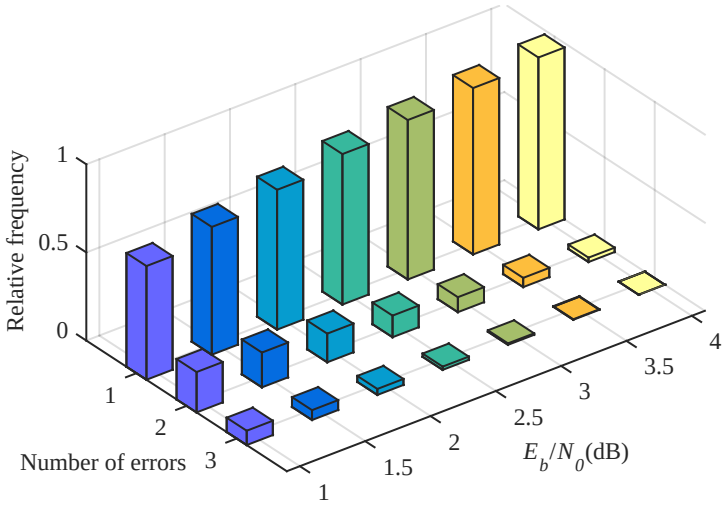


Figure 4. Frequency of errors caused by the channel noise at different E_b/N_0 for $P(64, 32)$, 1×10^5 polar codes.

3.1. Analysis of Error Propagation

In SC decoding, a polar code that is decided to be incorrect has multiple error bits. Figure 5 shows that the error bits caused by channel noise are much smaller than the error bits caused by error propagation. However, error propagation affects the BER, not the BLER. To improve the block error-correction performance, the error bits caused by the channel noise should be flipped. As observed in Figure 5, most of the error decisions are caused by the error propagation of the first error bit, thus it is required to construct a flipping set efficiently.

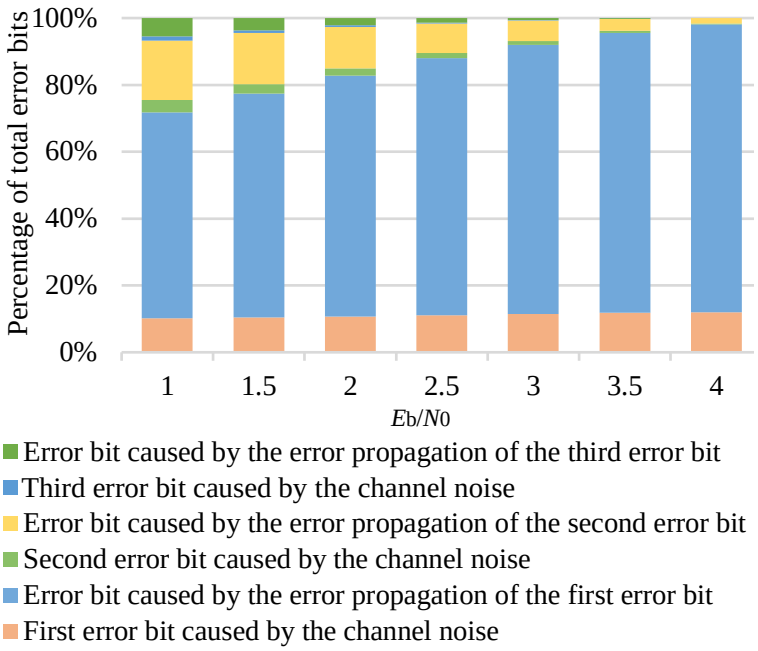


Figure 5. The percentage of error bits caused by different reasons at different E_b/N_0 for $P(64, 32)$, 1×10^5 polar codes.

3.2. DLSTM Network Structure

The first error bit plays an important role in the flipping algorithm. Since the first error bit cannot be accurately located by the LLR value, we adopt the neural network to predict it. A DLSTM network structure is illustrated in Figure 6b. This network mainly consists of an input layer, a DLSTM layer, and an output layer. The DLSTM layer contains $2N$ hidden

LSTM units, where N is the length of a polar code. The output layer is a fully connected layer with K neurons. This K represents the classification that the DLSTM network can recognize, as well as the length of the information bits. The K neurons in output layer activated by softmax function:

$$s(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_{j=0}^{K-1} e^{\hat{y}_j}} \quad , \tag{13}$$

where $\hat{y}_i$ is the output of the ith neuron. With this activation function, a K-dimensional vector is produced.

The input and output of the DLSTM network are:

- Input: The absolute value of the LLR sequence (both the information bits and the frozen bits) that fails CRC detector in the SC decoding.
- Output: A K-dimensional vector, the element of which corresponds to the probability of error occurrence for each information bit.

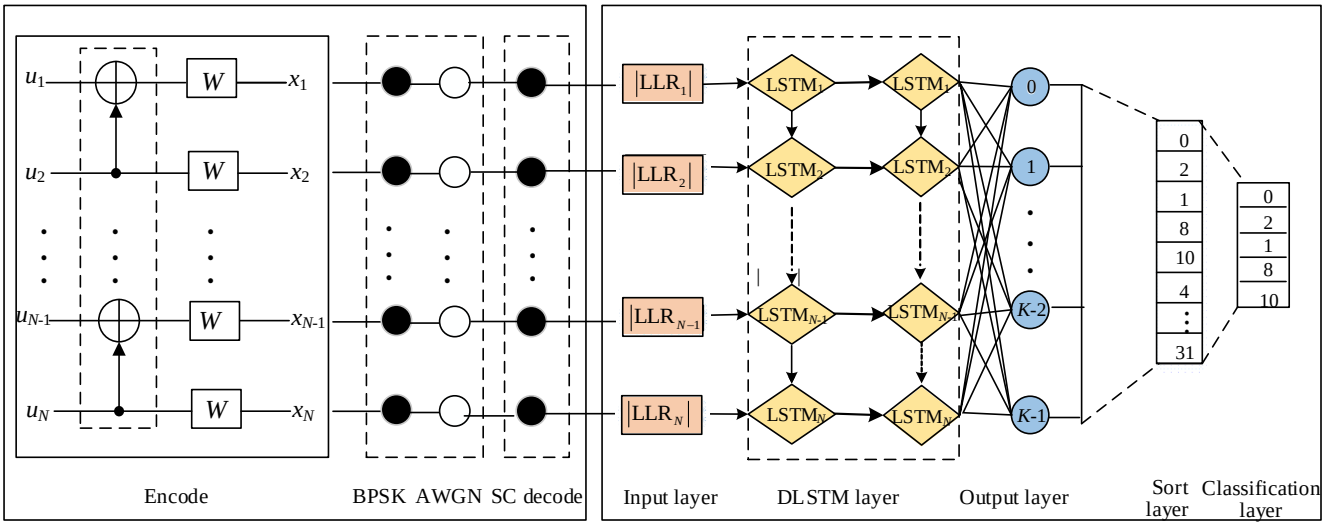


Figure 6. (**a**) System simulation model. (**b**) The DLSTM network structure for predicting the first error bit of $P(64, 32)$.

The details of the DLSTM unit are shown in Figure 7. At t time, the input of the DLSTM network is $[\mathbf{h}_{t-1}^1, \mathbf{x}_t]$, which represents the output $\mathbf{h}_{t-1}^1$ at $t-1$ time and the input $\mathbf{x}_t$ at the current time concatenated into a vector. The vector passes through the forget gate $\mathbf{f}_t^1$, the input gate $\mathbf{i}_t^1$, and the output gate $\mathbf{o}_t^1$ in turn. $\mathbf{h}_t^1$ and $\mathbf{h}_t^2$ denote the two outputs of the DLSTM unit. The cell states $\mathbf{c}_t^1$ and $\mathbf{c}_t^2$ hold the different long-term memories. The output gate $\mathbf{o}_t^1$ of the DLSTM has two directions, one is transmitted the $t+1$ moment and the other is the input of the next layer at t moment. Thus, the input to the second LSTM layer is $[\mathbf{h}_{t-1}^2, \mathbf{h}_t^1]$, which is different from the input to the first layer. The relationship between the inputs $[\mathbf{h}_{t-1}^2, \mathbf{h}_t^1]$ and outputs $\mathbf{h}_t^2$ of the second LSTM layer is described as follows:

$$\begin{cases} \mathbf{f}_t^2 = \sigma\left(\mathbf{W}_f^2 \cdot \left[\mathbf{h}_{t-1}^2, \mathbf{h}_t^1\right] + \mathbf{b}_f^2\right) \\ \mathbf{i}_t^2 = \sigma\left(\mathbf{W}_i^2 \cdot \left[\mathbf{h}_{t-1}^2, \mathbf{h}_t^1\right] + \mathbf{b}_i^2\right) \\ \mathbf{o}_t^2 = \sigma\left(\mathbf{W}_o^2 \cdot \left[\mathbf{h}_{t-1}^2, \mathbf{h}_t^1\right] + \mathbf{b}_o^2\right) \\ \mathbf{c'}_t^2 = \tanh\left(\mathbf{W}_c^2 \cdot \left[\mathbf{h}_{t-1}^2, \mathbf{h}_t^1\right] + \mathbf{b}_c^2\right) \\ \mathbf{c}_t^2 = \mathbf{f}_t^2 \circ \mathbf{c}_{t-1}^2 + \mathbf{i}_t^2 \circ \mathbf{c'}_t^2 \\ \mathbf{h}_t^2 = \mathbf{o}_t^2 \circ \tanh\left(\mathbf{c}_t^2\right) \end{cases} . \tag{14}$$

Due to this unit structure, the DLSTM network structure is more favorable than the traditional LSTM. Because of the DLSTM network's good memory characteristic, it can show superiority in extracting historical data features and can preserve long-term relevancy.

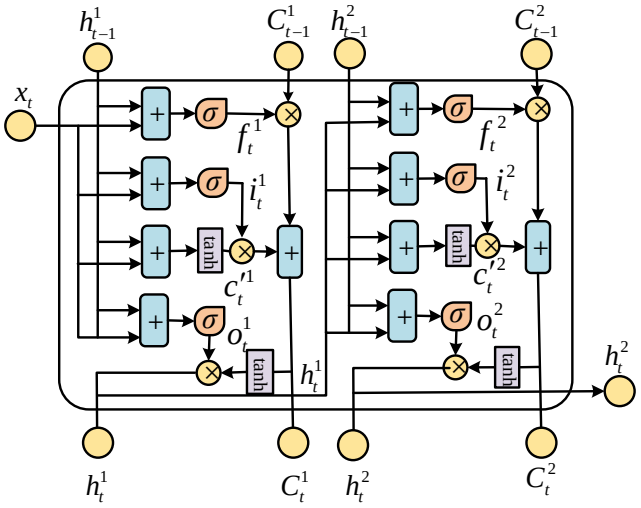


Figure 7. A DLSTM unit.

3.3. Training Process of the DLSTM network

To obtain the training data of the DLSTM network, we carried out the simulations presented in Figure 6a. First, we obtained the polar codes according to the method in Section 2. Then, we performed the binary phase shift keying (BPSK) demodulation and incorporated additive white Gaussian noise (AWGN) channel noise. After that, 2×10^6 polar codes were generated and decoded by the SC decoder. If the decoding result passed the CRC detector, the decoding was successful. Otherwise, it became the training sample for the DLSTM network. The sample consists of the absolute value of the LLR sequence and one-hot encoding of the first error bit, which is denoted as $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_j, \mathbf{y}_j)\}$, where j is the size of the sample, including the training set and validation set. $\mathbf{x}_c$, $1 \leq c \leq j$ is the input of the network. $\mathbf{y}_c$, $1 \leq c \leq j$ is the label and represents the expected output of the DLSTM network. The process of extracting information bits is shown in Table 1, where the binary sequence in the second row is the information pattern, in which '0' denotes the frozen bit and '1' denotes the information bit. The proposed algorithm clips all frozen bits in the output layer because flipping a frozen bit is useless and causes corresponding error propagation. The classification is reduced by $(N - K)/N$ compared to the method of Author17 [17].

Table 1. Extract information bits for $P(16, 8)$.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Location	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1
Index of information bit								0		1	2	3	4	5	6	7

In order to train a more efficient DLSTM network, the data were divided into the training set and validation set according to the ratio of 8:2. During the training, the regularization method [21] was used to prevent training overfitting. The L2-norm was applied to the weights, taking L2 = 0.008. The Adam optimizer [22] was adopted to design adaptive learning rates for different parameters because it can handle sparse gradients very well. The goal of the DLSTM network is to be a well-trained classifier where each information bit is a category. We employed the cross-entropy function to measure the loss between the expect value and the predicted value.

$$E(x) = -y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \ , \tag{15}$$

where y represents the labeled data and $\hat{y}$ represents the predicted value of the network. The smaller is the loss E, the better is the network trained. As the entire DLSTM is trained offline, it does not increase the SC flipping decoder complexity.

3.4. DLSTM-Based SC Flipping Algorithm

We utilized the well-trained DLSTM network to determine the flipping set of the first error bit. According to Author23 [23], the channels reliability ordering can be used to determine the flipping positions in SC flipping decoder. Therefore, we adopted channel reliability to determine the multiple error bits. This method avoids the ordering of the LLRs, which reduces the complexity of SC flipping decoder. Figure 8 illustrates the process of the DLSTM-based SC flipping algorithm, which consists of the following steps.

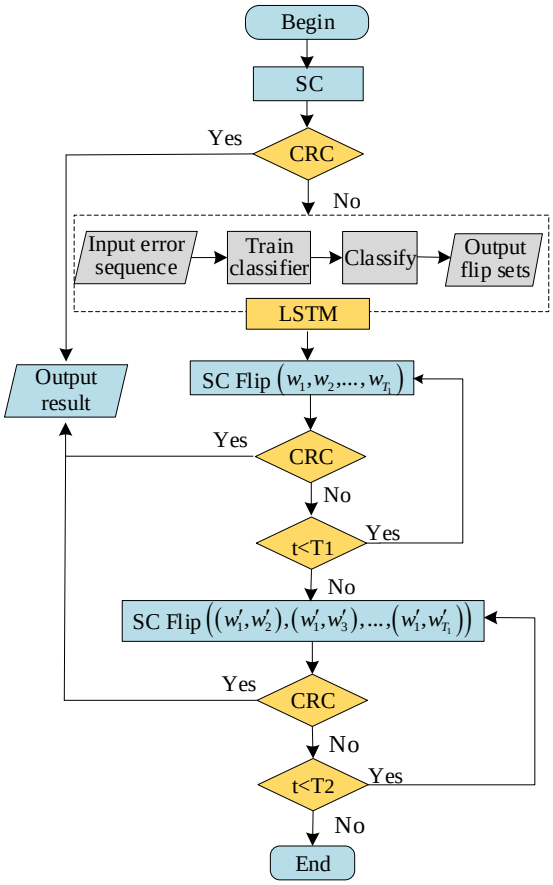


Figure 8. The SC Flipping algorithm based on DLSTM.

(1) Perform the SC decoding and check whether the decoding result can pass the CRC detector. If pass, the decoding is successful. Otherwise, go to Step 2.

(2) Take advantage of the well-trained DLSTM to predict the probability of error occurring in the information bits. Then, sort their probabilities in descending order. Select the index of the T_1 maximum probability and map them to the corresponding locations of the information bits. T_1 is the maximum number of one-bit flipping and is also the size of the flipping set. Employ these locations to construct the flipping set $\mathcal{W}_1^T = (w_1, w_2, \ldots, w_{T_1})$, where $w_1, w_2, \ldots, w_{T_1}$ are the candidate bits for the first error bit and the probability $w_1 > w_2 > \cdots > w_{T_1}$, and go to Step 3.

(3) Select w_1 for flipping. If the decoding fails, flip the following bits $w_2, w_3, \ldots, w_{T_1}$ successively. If T_1 attempts are unsuccessful, then go to Step 4. Otherwise, terminate the decoding.

(4) Calculate the channel reliability of flipping set $\mathcal{W}_1^T$ by Gaussian approximation. For $\mathcal{W}_1^T = (w_1, w_2, \ldots, w_{T_1})$, reorder into $\mathcal{W}_2^T = (w'_1, w'_2, \ldots, w'_{T_1})$ in ascending order

according to reliability, where the reliability follows: $w'_1 < w'_2 < \cdots < w'_{T_1}$. Select w'_1 and w'_2 from $\mathcal{W}_2^T$ to perform bit flipping at the same time. If the CRC detector fails, next select (w'_1, w'_3), (w'_1, w'_4), …, (w'_{T_1-1}, w'_{T_1}) in turn to flip. If T_2 attempts are unsuccessful, the decoding fails, where T_2 is the maximum number of two-bit flipping. Otherwise, the decoding is successful. The selection method is shown in Figure 9.

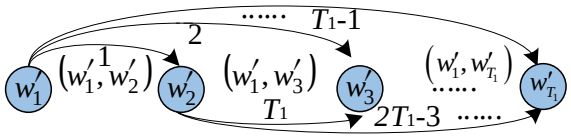


Figure 9. Diagram of two-bit flipping selection.

The pseudocode for two-bit flipping algorithm based on the DLSTM network is shown in Algorithm 2. It can be extended to multi-bit flipping in a similar way. The maximum number of candidate bits is K. The set of multi-bit flipping is denoted as $\mathcal{W}_3^T, \mathcal{W}_4^T, \ldots, \mathcal{W}_K^T$. When simultaneously flipping the least reliable 3, 4, …, K bits, the maximum number of flips is indicated as $T_3, T_4, \ldots, T_K$.

Algorithm 2 Two-bit flipping algorithm based on the DLSTM.

Input: $y_1^N, \mathcal{W}_1^T, \mathcal{W}_2^T, T_1, T_2$
Output: $\hat{u}_1^N$

$\{\hat{u}_1^N$,LLR$\} \leftarrow$ SC (y_1^N)
$t_1 = 0, t_2 = 0$
while $\hat{u}_1^N$ fail CRC and $t_1 < T_1$ **do**
  $\mathcal{W}_1^T \leftarrow$ DLSTM
  $\hat{u}_1^N \leftarrow$ SC flip $(y_1^N, \mathcal{W}_1^T(t_1))$
  if $\hat{u}_1^N$ pass CRC **then**
    break
  else
    $t_1 \leftarrow t_1 + 1$
  end if
end while
if $t_1 \geq T_1$ **then**
  $t_1 = 0$
  while $\hat{u}_1^N$ fail CRC **do**
    for $i \leftarrow t_1 + 1$ to T_1 **do**
      if $t_2 < T_2$ **then**
        $\hat{u}_1^N \leftarrow$ SC flip $(y_1^N, \mathcal{W}_2^T(t_1, i))$
        if $\hat{u}_1^N$ pass CRC **then**
          break
        else
          $t_2 \leftarrow t_2 + 1$
        end if
      end if
    end for
    $t_1 \leftarrow t_1 + 1$
  end while
end if
return $\hat{u}_1^N$

3.5. DLSTM-Based Robustness Mechanism

Current the neural network decoders are specific, such as certain block length or signal-to-noise ratios (SNR). To make the algorithm robust, we designed a DLSTM network architecture that is compatible with multiple block lengths. For different block length

polar codes $P_1(N_1, K_1)$, $P_2(N_2, K_2)$, $P_3(N_3, K_3)$, and $N_1 > N_2 > N_3$, we make the following updates to maintain the integrity of the input shape. The length N_1 of the longest codewords is taken as the standard, and padding is performed for codewords smaller than this length. One-hot encoding is done directly for labels with length K_1. After making the length uniform for polar codes with variable length, the padding part is learned during the training, which affects the prediction results. To reduce the impact, a mask layer is added before the DLSTM layer for skipping the respective time step. During the DLSTM training, the part that is masked is not involved in the calculation. Figure 10 presents this training process. To obtain better generalization capability, we adopt an early stopping criterion. The training is stopped if the accuracy of the validation set does not improve.

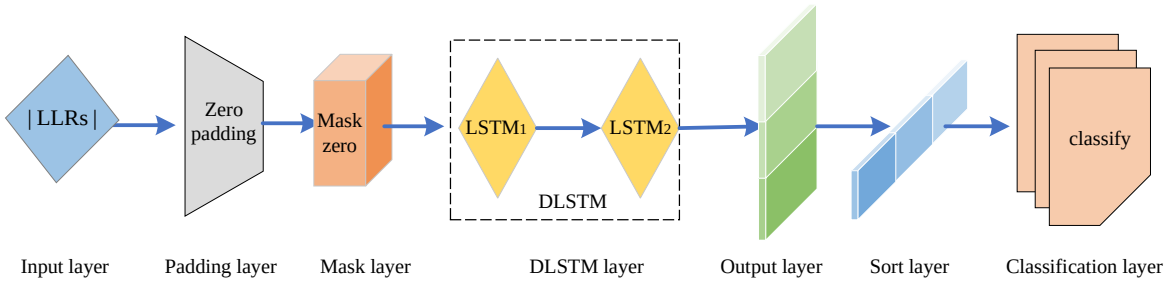


Figure 10. Structure of the DLSTM-based robustness algorithm.

4. Performance Analysis

In this section, we first give the DLSTM network training results. Then, we evaluate the decoding delay, complexity, and error-correction performance of the proposed algorithm and other existing algorithms. Simulations were performed with AWGN channel and BPSK modulation on an Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz server.

4.1. DLSTM Network Training Results

The DLSTM network was trained by the LLR sequences and the one-hot encoding of first error bit. The simulation parameters for $P(64, 32)$ are shown in Table 2. As described in Section 2, a basic LSTM unit consists of three gates and one cell state. Each gate and cell state is a fully connection layer with N (hidden size) neurons. Thus, a layer of LSTM actually has four fully connection layers. Equation (9) can also be written as

$$\text{gate}(\mathbf{x}_t) = \sigma(\mathbf{W} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}) \ . \tag{16}$$

Therefore, one layer of the basic LSTM network has $4(h_s(h_s + x_d)+h_s)$ parameters, where h_s is the size of the hidden layer and x_d is the dimension of the input data. As shown in Table 2, a layer of basic LSTM has $4 \times [64 \times (64 + 64)+64]$ = 33,024 parameters. Thus, a layer of DLSTM has $2\times$ 33,024 = 66,048 parameters. Dense denotes the fully connection layer. The N(hidden size)-dimensional vector output from the DLSTM network is used as the input to the dense layer, so the dense layer has $(h_s \cdot K + K)$ parameters, where K is the number of neurons in the dense layer.

All the data are trained once for one epoch. In each epoch, the network calculates the accuracy and loss of the training set and validation set. After 30 epochs, the accuracy and loss changes for different codewords with E_b/N_0 = 1.0 dB are depicted in Figure 11. It can be seen that, before five epochs, the accuracy of the validation set is higher than that of the training set. As the number of epoch increases, the accuracy of the training set starts to increase and outperform that of the validation set. Finally, the accuracy of both the validation set and training set converge to a stable value. It is observed that the DLSTM network converges and reaches a stable state after about 20 epochs.

In this paper, a flipping set of size T_1 is constructed for one-bit flipping. Figure 12 shows that, when the number of flips is smaller than the size of the CS, our flipping set has a higher accuracy in identifying the first error bit compared to the CS in [8]. Although CS

contains the first incorrect hard decision with high probability, the simulation results show that superior error correction performance is achieved when the number of flips is comparable to the size of CS. Figure 13 also illustrates the comparison of identifying the first error bit at E_b/N_0 = 1.0 dB. The proposed algorithm has higher accuracy than the other two algorithms within the same number of flips. At the first flip, the accuracy of the proposed algorithm is improved by 2.5% and 9.7% compared to the DL-based algorithm and the DSCF decoder, respectively. Table 3 displays the comparison of the two- and three-layer LSTMs. The accuracy of the two-layer network in predicting the first error bit is comparable, but the training parameters of the DLSTM network are reduced by about 1/3. Therefore, it is appropriate to train with a DLSTM network, which can effectively reduce the training complexity.

Table 2. Simulation parameter for $P(64, 32)$ with E_b/N_0 = 1.0 dB.

Name	Parameter
Polar codes	(64,32)
Frame number	2,000,000
Rate R	1/2
CRC generator polynomial	$x^8 + x^7 + x^5 + x^4 + x^1 + 1$
Batch size	1000
Number of epoch	30
Regularization L2	0.008
DLSTM	66,048
Dense	2080
Optimizer	Adam

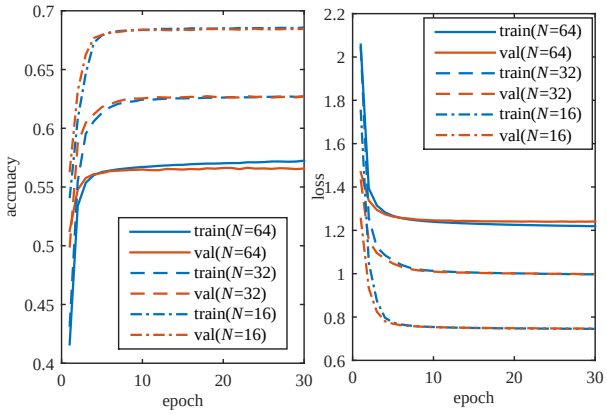


Figure 11. Convergence graph for different codewords with E_b/N_0 = 1.0 dB.

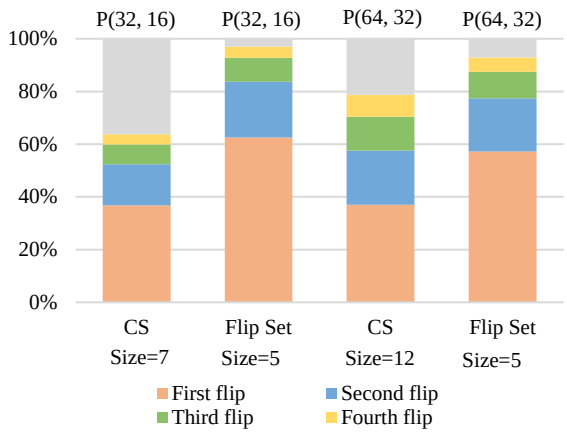


Figure 12. A comparison of the CS in [8] and the flipping set in this paper for identifying the first error bit at E_b/N_0 = 1.0 dB.

Table 3. Comparison of two- and three-layer LSTM networks in predicting the first error bit.

Polar Codes	Layer	Accuracy				Total Param
$P(64,32)$	2	57.2%	77.45%	87.4%	92.84%	68,128
	3	57.12%	77.32%	87.43%	92.94%	101,152
$P(16,8)$	2	68.63%	91.8%	98.48%	99.86%	4360
	3	68.46%	91.81%	98.44%	99.85%	6472

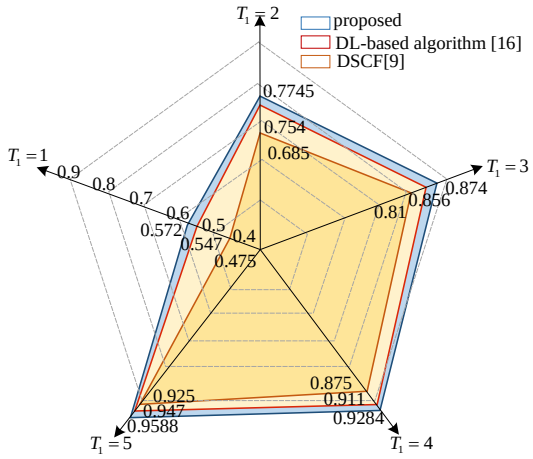


Figure 13. Accuracy comparison of identifying the first error bit for $P(64,32)$ at $E_b/N_0 = 1.0$ dB.

4.2. Decoding Complexity and Latency Analysis

The SC flipping decoder starts with the SC decoder. The complexity of SC is $O(N \log N)$. In constructing the flipping set, the complexity of the sorting operation on K probabilities is $O(K \log K)$. When flipping one bit, there are no more than T_1 additional attempts, and the complexity is $O(T_1 N \log N)$. When flipping two bits, the T_1 candidate bits are sorted according to channel reliability with a complexity of $O(T_1 \log T_1)$. Then, no more than T_2 additional attempts are performed, and the complexity is $O(T_2 N \log N)$. Finally, if $T_1 > T_2$, the maximum complexity of the proposed algorithm is $O(T_1 N \log N)$. Otherwise, the maximum complexity is $O(T_2 N \log N)$. However, the flipping is only for the error sequence that fails the CRC detector, so the average complexity is related to the BLER. Therefore, the average complexity of the proposed algorithm is $O((1 + P_e(T_1 + T_2))N \log N + K \log K + T_1 \log T_1)$, where P_e is the BLER of the SC decoding. As shown in Figure 14, the average decoding complexity is high because P_e is dramatically large at low E_b/N_0. However, P_e decreases rapidly as E_b/N_0 increases, and the average complexity rapidly decreases, finally converging to the complexity of the SC decoding. It can be noticed that the average decoding complexity is lower than the SCL decoding ($L = 4$) and the DSCF decoding and slightly higher than the ML-MSCF decoding.

It takes a lot of time to train an effective DLSTM network, but the whole training is done offline and can benefit from some acceleration platforms [24]. Therefore, the training process does not increase the decoding latency. However, the prediction time is calculated as part of the total decoding latency. Compared with the traditional approach, the prediction is an additional computational process. In Figure 15, the decoding latency represents the total time consumed for decoding as calculated by the CPU. The proposed algorithm consumes comparable time to the ML-MSCF decoder. When $E_b/N_0 = 1.0$ dB, the consumed time is decreased by 11.41% and 27.68% compared to the DSCF decoder for both code lengths, which saves a lot of computation time by avoiding exponential and logarithmic operations, respectively. Compared to CA-SCL ($L = 2$), the consumed time is lowered by 3.64% at $N = 32$ but comparable at $N = 64$. Compared with CA-SCL ($L = 4$), the time consumption is reduced by 20.32% and 31.26%, respectively, and the time to try L decoding paths is reduced.

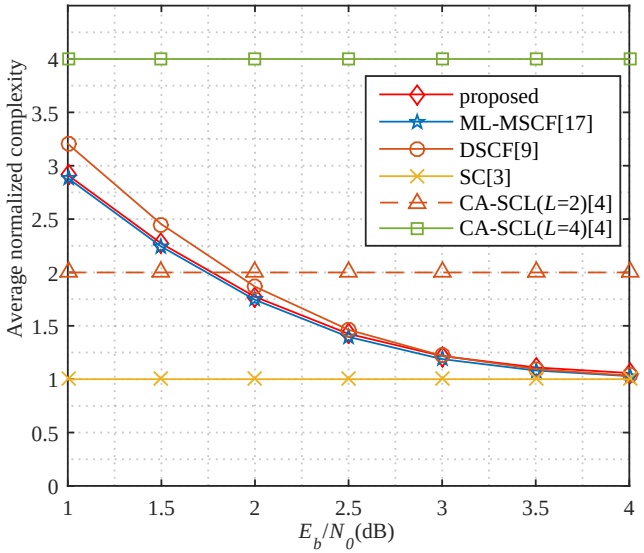


Figure 14. Comparison of average normalized decoding complexity for $P(64, 32)$ with $T_1 = 4, T_2 = 2$.

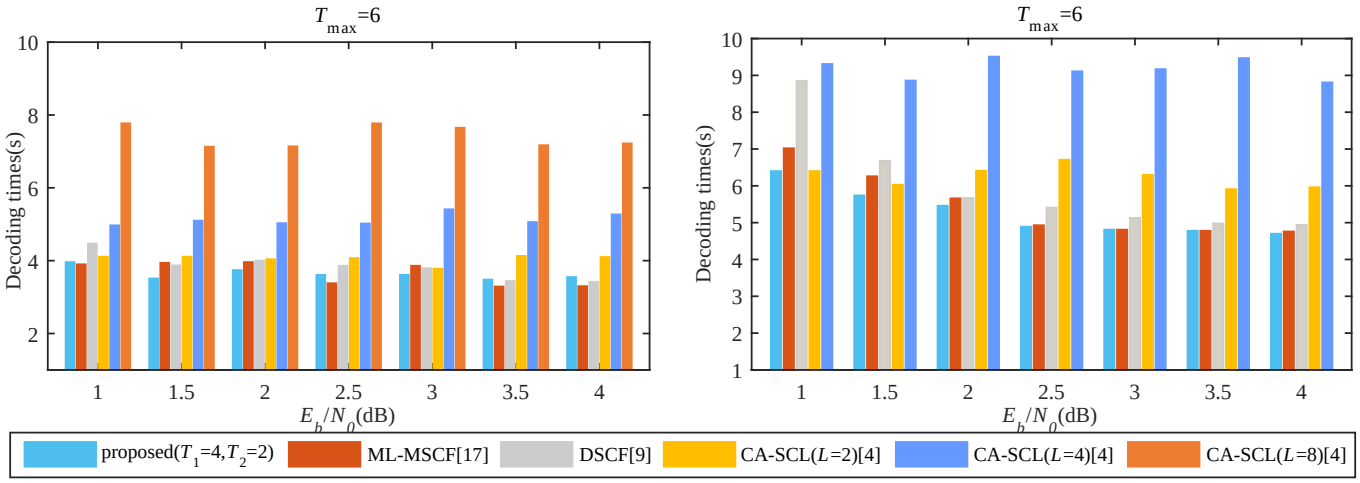


Figure 15. (**left**) Comparison of total decoding latency for $P(32, 16)$. (**right**) Comparison of total decoding latency for $P(64, 32)$.

4.3. BER and BLER Analysis

In Figure 16 left, we compare the error-correction performance for the proposed algorithm with $T_1 = 4$, $T_2 = 2, 4$, and $T_3 = 4$ for $P(32, 16)$. The proposed decoder adopts a 4-bit CRC and its generator polynomial is $g(x) = x^4 + x + 1$. It can be noticed that, with continuous flipping, a lot of gain can be achieved. When BLER = 10^{-3}, the proposed algorithm with $T_1 = 4$, $T_2 = 2$, and $T_1 = 4$, $T_2 = 4$ can achieve gains of approximately 0.32 dB and 0.57 dB than that with $T_1 = 4$, respectively. At BER = 10^{-4}, the proposed algorithm with $T_1 = 4$, $T_2 = 2$, and $T_1 = 4$, $T_2 = 4$ can obtain performance improvement of about 0.25 and 0.46 dB compared to that with $T_1 = 4$, respectively. The error-correction performance of the proposed algorithm with $T_1 = 4$ and $T_2 = 4$ is close to that with $T_1 = 4$, $T_2 = 4$, and $T_3 = 4$. This is due to the occurrence of three errors with low frequency. In Figure 16 right, the block length is changed to 64. The proposed algorithm with $T_1 = 4$ and $T_2 = 4$ is lower than that with $T_1 = 4$, $T_2 = 2$, and $T_1 = 4$ in terms of the BLER and the BER. The proposed algorithm for $N = 32$ has a significant performance gain compared to that for $N = 64$. On the one hand, the shorter is the block length, the higher is the prediction accuracy. On the other hand, the degree of channel polarization for $N = 64$ is better than for $N = 32$. A polar code with $N = 64$ has fewer error bits than that with $N = 32$.

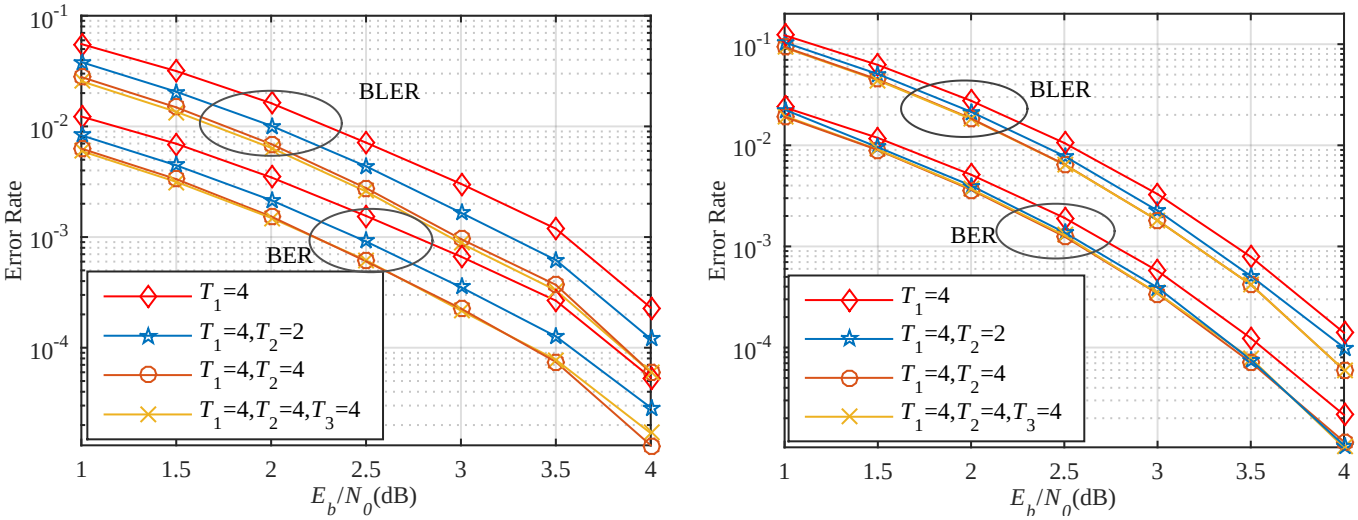


Figure 16. (**left**) Performance comparison of different flip times for $P(32, 16)$. (**right**) Performance comparison of different flip times for $P(64, 32)$.

Figure 17 left shows the BLER performance of the proposed algorithm compared with the existing algorithms for $P(32, 16)$ with $T_{\max} = 6$. It can be seen that the proposed algorithm significantly outperforms the ML-MSCF decoder and the DSCF decoder within the same number of flips. It is also more efficient than CA-SCL ($L = 2$, $L = 4$). It is close to the performance of CA-SCL ($L = 8$). As shown in Table 4, the proposed algorithm can achieve approximately 0.28 and 0.49 dB performance gain, but the decoding latency is increased by 5.77% and 1.16% over the ML-MSCF decoder and the DSCF decoder at BLER = 10^{-3}. Compared with CA-SCL ($L = 2$, $L = 4$), it can obtain performance gain of about 0.66 and 0.19 dB, and the decoding latency is reduced by 15.73% and 31.22%, respectively. The performance is slightly lower than that of CA-SCL ($L = 8$), but the decoding latency is decreased by 51.46%. In Figure 17 right, compared with the ML-MSCF decoder, DSCF decoder, and CA-SCL ($L = 2$, $L = 4$), the proposed algorithm has lower BER.

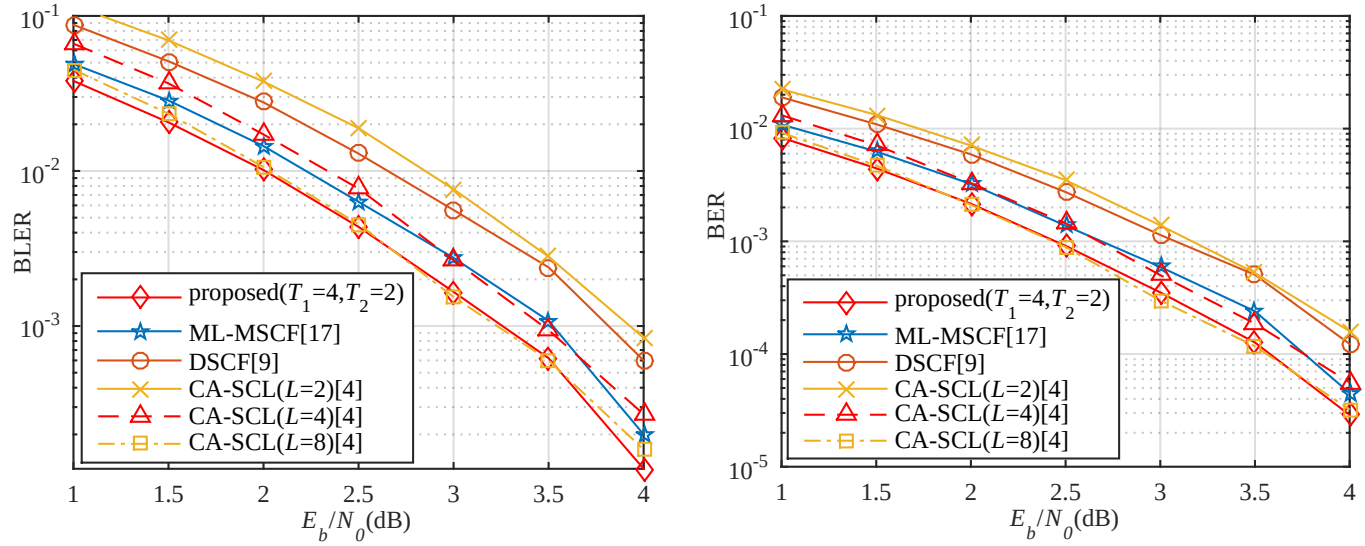


Figure 17. (**left**) BLER comparison of different algorithms for $P(32, 16)$. (**right**) BER comparison of different algorithms for $P(32, 16)$.

Table 4. The obtained gain and reduced latency by the proposed algorithm for $P(32, 16)$ with $T_{\max} = 6$ and BLER = 10^{-3}.

Algorithm	E_b/N_0	Gain (dB)	Reduce Latency
proposed	3.25	-	-
ML-MSCF [17]	3.53	0.28	−5.77%
DSCF [9]	3.74	0.49	−1.16%
CA-SCL(L = 2) [4]	3.91	0.66	15.73%
CA-SCL(L = 4) [4]	3.44	0.19	31.22%
CA-SCL(L = 8) [4]	3.23	−0.02	51.46%

Figure 18 compares the BLER performance for $P(64, 32)$ with $T_{\max} = 6$. It can be noticed that the proposed algorithm has better error-correction performance than the ML-MSCF decoder, DSCF decoder, and CA-SCL (L = 2). It is close to the performance of CA-SCL (L = 4). Table 5 shows the obtained gains and reduced latency using the proposed algorithm over other algorithms at BLER = 10^{-3}. The proposed algorithm achieves a gain of about 0.21 dB compared to the ML-MSCF decoder, but the decoding latency is slightly increased. Compared with the DSCF decoder and the CA-SCL (L = 2) decoder, the proposed algorithm can obtain gains of about 0.36 and 0.55 dB, and decoding latency is reduced by 1.39% and 19.29%, respectively. The performance is slightly lower than that of CA-SCL (L = 4), but the decoding latency is decreased by 45.41%. This is due to multiple additional attempts when flipping one bit and selecting the most least reliable position for multi-bit flipping based on channel reliability. When the code length is 32, the proposed algorithm can approach the performance of CA-SCL (L = 8), but, when the code length is 64, it is close to the performance of CA-SCL (L = 4). This is because the accuracy of the proposed algorithm for $N = 32$ is higher than that for $N = 64$.

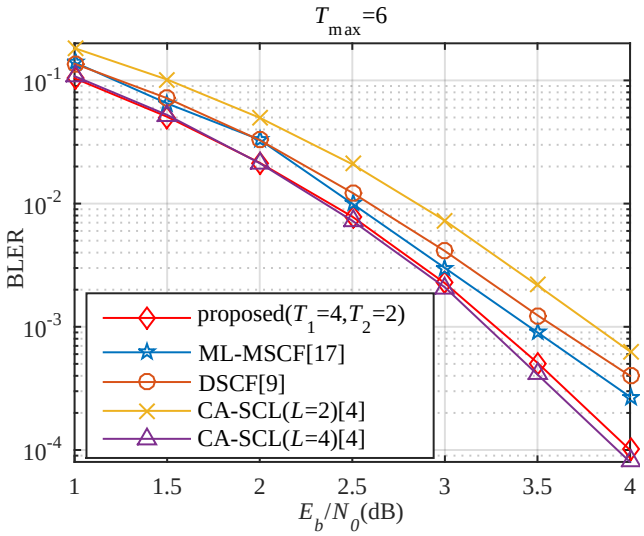


Figure 18. BLER comparison of different algorithms for $P(64, 32)$.

Table 5. The obtained gain and reduced latency by the proposed algorithm for $P(64, 32)$ with $T_{\max} = 6$ and BLER = 10^{-3}.

Algorithm	E_b/N_0	Gain (dB)	Reduce Latency
proposed	3.26	-	-
ML-MSCF [17]	3.47	0.21	−0.62%
DSCF [9]	3.62	0.36	1.39%
CA-SCL(L = 2) [4]	3.81	0.55	19.29%
CA-SCL(L = 4) [4]	3.24	−0.02	45.41%

4.4. BLER Analysis of the Algorithm with Robustness Mechanism

The proposed algorithm with robustness mechanism is compatible with different block lengths ($N_1 = 64$, $N_2 = 32$, and $N_3 = 16$). The proposed decoder for $N_3 = 16$ with a 2-bit CRC has a generator polynomial of $g(x) = x^2 + 1$. Figure 19 demonstrates that the BLER performance of the proposed algorithm and optimized algorithm is close for the three block lengths. The optimized algorithm has generalization ability and can break the block length constraint during neural network training. The large-scale processing capability of the neural network is also exploited.

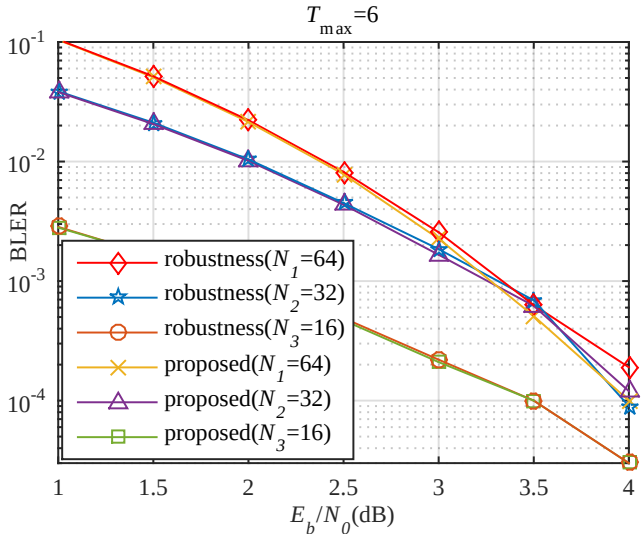


Figure 19. BLER comparison of the proposed algorithm and optimized algorithm with robustness mechanism with $T_{\max} = 6$.

5. Conclusions

To improve the performance of short block length, this paper proposes a DLSTM-based SC flipping algorithm for short polar codes. The DLSTM is used to predict the first error bit and construct a flipping set for the one-bit flipping, clipping all frozen bits to enhance the accuracy of prediction. For the sequences that fail CRC detector, the flipping set is resorted according to channel reliability, and the most unreliable bits are selected for multi-bit flipping. The simulation results show that the proposed algorithm has better error-correction performance compared with the ML-MSCF decoder and the DSCF decoder. In addition, when E_b/N_0 = 1.0 dB, the decoding performance is close to CA-SCL (L = 8) for N = 32, and the decoding latency is reduced by 49.03%. Furthermore, to make the algorithm robust, we propose a mechanism to make it compatible with different block lengths. For future work, we will focus on multi-task neural networks to predict multiple error bits of polar codes simultaneously.

Author Contributions: Conceptualization, J.C., W.K. and X.Z.; methodology, W.K. and X.Z.; software, W.K. and X.Z.; validation, J.C., W.K. and X.Z.; formal analysis, W.K. and X.Z.; investigation, J.C. and X.Z.; resources, J.C. and W.K.; data curation, W.K. and X.Z.; writing—original draft preparation, J.C., W.K. and X.Z.; writing—review and editing, D.C. and Q.Z.; visualization, X.Z.; supervision, Q.Z.; and project administration, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Joint Fund for Smart Computing of Natural Science Foundation of Shandong Province (No. ZR2019LZH001) and the Shandong University Youth Innovation Supporting Program (Nos. 2019KJN020 and 2019KJN024).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no known competing financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Abbreviations

The following abbreviations are used in this manuscript:

5G	Fifth generation
SC	Successive cancellation
DLSTM	Double long short term memory
CRC	Cyclic redundancy check
LLR	Log-likelihood ratio
CS	Critical set
BER	Bit error rate
BLER	Block error rate
BPSK	Binary phase shift keying
AWGN	Additive white Gaussian noise

References

1. Arıkan, E. Channel polarization: A method for constructing capacity-achieving codes. In Proceedings of the 2008 IEEE International Symposium on Information Theory, Toronto, ON, Canada, 6–11 July 2008; pp. 1173–1177.
2. Final Report of 3GPP TSG RAN WG1 #87 v1.0.0; Reno, USA. November 2016. Available online: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_87/Report/Final_Minutes_report_RAN1%2387_v100.zip (accessed on 5 January 2021).
3. Arıkan, E. Channel Polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073. [CrossRef]
4. Niu, K.; Chen, K. CRC-aided decoding of polar codes. *IEEE Commu. Lett.* **2012**, *16*, 1668–1671. [CrossRef]
5. Tal, I.; Vardy, A. How to construct polar codes. *IEEE Trans. Inf. Theory* **2013**, *59*, 6562–6582. [CrossRef]
6. Afisiadis, O.; Balatsoukas-Stimming A.; Burg, A. A low-complexity improved successive cancellation decoder for polar codes. In Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2–5 November 2014; pp. 2116–2120.
7. Chandesris, L.; Savin, V.; Declercq, D. An improved SCFlip decoder for polar codes. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
8. Zhang, Z.; Qin, K.; Zhang, L.; Chen, G.T. Progressive bit-flipping decoding of polar codes over layered critical sets. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Singapore, 4–8 December 2017; pp. 4–8.
9. Chandesris, L.; Savin, V.; Declercq, D. Dynamic-SCFlip decoding of polar codes. *IEEE Trans. Commu.* **2018**, *66*, 2333–2345. [CrossRef]
10. Doan, N.; Ali Hashemi, S.; Gross, W.J. Neural successive cancellation decoding of polar codes. In Proceedings of the 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Kalamata, Greece, 25–28 June 2018; pp. 1–5.
11. Doan, N.; Hashemi, S.A.; Ercan, F.; Tonnellier, T.; Gross, W.J. Neural dynamic successive cancellation flip decoding of polar codes. In Proceedings of the 2019 IEEE International Workshop on Signal Processing Systems (SiPS), Nanjing, China, 20–23 October 2019; pp. 272–277.
12. Dhok, A.; Bhole, S. ATRNN: Using seq2seq approach for decoding polar codes. In Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 7–11 January 2020; pp. 662–665.
13. Qin, Y.; Liu, F. Convolutional neural network-based polar decoding. In Proceedings of the 2019 2nd World Symposium on Communication Engineering (WSCE), Nagoya, Japan, 20–23 December 2019; pp. 189–194.
14. Wen, C.; Xiong, J.; Gui, J.; Shi, J.; Wang, Y. A novel decoding scheme for polar code using convolutional neural network. In Proceedings of the 2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Jeju, Korea, 5–7 June 2019; pp. 1–5.
15. Chen, C.; Teng, C.; Wu, A. Low-Complexity LSTM-assisted bit-flipping algorithm for successive cancellation list polar decoder. In Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 1708–1712.
16. Wang, X.; Zhang, H. Learning to flip successive cancellation decoding of polar codes with LSTM networks. In Proceedings of the 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Istanbul, Turkey, 8–11 September 2019; pp. 1–5.
17. He, B.; Wu, S.; Deng, Y.; Yin, H.; Jiao, H.; Zhang, Q. A machine learning based multi-flips successive cancellation decoding scheme of polar codes. In Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 25–28 May 2020; pp. 1–5.

18. Kasongo, S.M.; Sun, Y. A Deep Long Short-Term Memory based classifier for Wireless Intrusion Detection System. *ICT Express* **2020**. [CrossRef]
19. Wen, G.; Qin, J.; Fu, X.; Yu, W. DLSTM: Distributed Long Short-Term Memory Neural Networks for the Internet of Things. *IEEE Trans. Netw. Sci. Eng.* **2021**. [CrossRef]
20. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comp.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
21. Yang, H.; Ma, J.H. Research on method of regularization parameter solution. *Comput Meas. Control* **2017**, *25*, 226–229.
22. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
23. Zhang, Z.; Niu, K.; Dong, C. Multi-bit-flipping decoding of polar codes based on medium-level bit-channels sets. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–6.
24. Shantharama, P.; Thyagaturu, A.S.; Reisslein, M. Hardware-Accelerated Platforms and Infrastructures for Network Functions: A Survey of Enabling Technologies and Research Studies. *IEEE Access* **2020**, *8*, 132021–132085. [CrossRef]