Journal of Signal Processing Systems (2021) 93:631–642
https://doi.org/10.1007/s11265-020-01599-y

Neural Successive Cancellation Flip Decoding of Polar Codes

Nghia Doan[1] · **Seyyed Ali Hashemi**[2] · **Furkan Ercan**[1] · **Thibaud Tonnellier**[1] · **Warren J. Gross**[1]

Received: 15 April 2020 / Revised: 13 August 2020 / Accepted: 23 September 2020 / Published online: 22 October 2020

Abstract

Dynamic successive cancellation flip (DSCF) decoding of polar codes is a powerful algorithm that can achieve the error correction performance of successive cancellation list (SCL) decoding, with an average complexity that is close to that of successive cancellation (SC) decoding at practical signal-to-noise ratio (SNR) regimes. However, DSCF decoding requires costly transcendental computations to calculate a bit-flipping metric, which adversely affect its implementation complexity. In this paper, we first show that a direct application of common approximation schemes on the conventional DSCF decoding results in a significant error-correction performance loss. We then introduce an additive perturbation parameter and propose an approximation scheme which completely removes the need to perform transcendental computations in DSCF decoding. Machine learning (ML) techniques are then utilized to optimize the perturbation parameter of the proposed scheme. Furthermore, a quantization scheme is developed to enable efficient hardware implementation. Simulation results show that when compared with DSCF decoding, the proposed decoder with quantization scheme only experiences a negligible error-correction performance degradation of less that 0.08 dB at a target frame-error-rate (FER) of 10^{-4}, for a polar code of length 512 with 256 information bits. In addition, the bit-flipping metric computation of the proposed decoder reduces up to around 31% of the number of additions used by the bit-flipping metric computation of DSCF decoding, without any need to perform costly transcendental computations and multiplications.

Keywords 5G · Polar codes · Deep learning · SC flip

1 Introduction

Polar codes are proven to achieve channel capacity for any binary symmetric channel under the low-complexity successive cancellation (SC) decoding as the code length increases towards infinity [3]. Recently, polar codes are selected for use in the enhanced mobile broadband (eMBB) control channel of the fifth generation of cellular mobile communications (5G standard), which requires codes of short length [1]. The error-correction performance of SC decoding for short polar codes does not satisfy the requirements of the 5G standard. A SC list (SCL) decoding was introduced in [22] to improve the error-correction performance of SC decoding for short to moderate polar codes by maintaining a list of candidate codewords at each decoding step. In addition, it was observed that under SCL decoding, the error-correction performance is significantly enhanced when the polar code is concatenated with a cyclic redundancy check (CRC) [22]. However, the implementation complexity of SCL decoding grows as the list size increases [10, 13, 14].

SC flip (SCF) decoding algorithm was introduced in [2]. Unlike SCL decoding, SCF decoding performs multiple SC decoding attempts in series, where each decoding attempt tries to flip the first-order erroneous information bit of the previous decoding attempt. Similar to SCL decoding, SCF

✉ Nghia Doan
nghia.doan@mail.mcgill.ca

Seyyed Ali Hashemi
ahashemi@stanford.edu

Furkan Ercan
furkan.ercan@mail.mcgill.ca

Thibaud Tonnellier
thibaud.tonnellier@mcgill.ca

Warren J. Gross
warren.gross@mcgill.ca

1 Department of Electrical and Computer Engineering, McGill University, Montreal, Canada

2 Department of Electrical Engineering, Stanford University, Stanford, USA

decoding relies on a CRC to indicate whether the decoding is successful or not. Several methods have been recently proposed to improve the error-correction performance of SCF decoding [6, 9, 11]. However they are limited with correcting a single erroneous bit in the codeword. Dynamic SCF (DSCF) decoding [5] is a generalization of SCF-based decoding that is able to correct multiple erroneous bits under SC decoding [5].

The advantage of DSCF decoding is that the average decoding complexity of it at high signal-to-noise ratio (SNR) regimes asymptotically approaches the decoding complexity of SC decoding, while maintaining an error-correction performance comparable to that of SCL decoding [5]. However, DSCF decoding requires costly exponential and logarithmic computations that prevent the algorithm to be attractive for practical applications.

In this paper, we first show that a direct application of the common approach to approximate the underlying exponential and logarithmic function in the DSCF decoding algorithm results in a significant error-correction performance degradation. We then introduce a new trainable perturbation parameter to the DSCF decoding algorithm and show that the proposed decoder does not suffer from significant error-correction performance loss if the common hardware-friendly approximation techniques are used. Unlike DSCF decoding, where the parameter is optimized using a Monte-Carlo simulation, the parameter optimization of the proposed decoder is formalized as a classification problem and is solved using machine learning (ML) techniques. Thus, we name the proposed decoder as neural SCF (NSCF) decoding. In addition, the symmetric property of the proposed decoder is utilized to simplify the training process. Furthermore, to address hardware implementation, parameter quantization is considered during the training process.

Simulation results show that compared to the state-of-the-art DSCF decoding, both full-precision and quantized schemes of the proposed NSCF decoder experience a negligible error-correction performance degradation of less than 0.08 dB at a target frame-error-rate (FER) of 10^{-4} for a 5G polar code of length 512, with 256 information bits, concatenated with a 24-bit 5G CRC. The proposed bit-flipping metric computation reduces up to 31% of the number of additions required by that of DSCF decoding, while completely removing the need to perform costly transcendental computations and multiplications as is required in DSCF decoding.

This paper is an extension of the work in [7]. Compared to [7], an efficient training algorithm is introduced to significantly reduce the number of training data. In addition, a quantization scheme is considered during the parameter optimization process that targets a hardware implementation of the proposed decoder.

The rest of this paper is organized as follows. Section 2 briefly reviews polar codes and DSCF decoding. Section 3 describes the proposed NSCF decoder. Section 4 provides numerical results, and finally, Section 5 presents concluding remarks.

2 Preliminaries

2.1 Polar Codes

A polar code $\mathcal{P}(N, K)$ of length N with K information bits is constructed by applying a linear transformation to the message word $\boldsymbol{u} = \{u_0, u_1, \ldots, u_{N-1}\}$ as $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}^{\otimes n}$, where $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ is the codeword, $\boldsymbol{G}^{\otimes n}$ is the n-th Kronecker power of the polarizing matrix $\boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, and $n = \log_2 N$. The vector $\boldsymbol{u}$ contains a set $\mathcal{A}$ of K information bits and a set $\mathcal{A}^c$ of $N - K$ frozen bits. The positions of the frozen bits are known to the encoder and the decoder and their values are usually set to 0. The codeword $\boldsymbol{x}$ is then sent through the channel using binary phase-shift keying (BPSK) modulation. The soft vector of the transmitted codeword received by the decoder is $\boldsymbol{y} = (\mathbf{1} - 2\boldsymbol{x}) + \boldsymbol{z}$, where $\mathbf{1}$ is an all-one vector of size N, and $\boldsymbol{z} \in \mathbb{R}^N$ is the additive white Gaussian noise (AWGN) vector with variance σ^2 and zero mean. In the log-likelihood ratio (LLR) domain, the LLR vector of the transmitted codeword is

$$\boldsymbol{L}_n = \frac{2\boldsymbol{y}}{\sigma^2}. \tag{1}$$

2.2 Successive Cancellation Decoding

SC decoding can be illustrated on a polar code factor graph representation. An example of a factor graph for $\mathcal{P}(8, 5)$ is depicted in Figure 1a. To obtain the message word, the soft LLR values and the hard bit estimations are propagated through all the processing elements (PEs), which are depicted in Figure 1b. Each PE performs the following computations

$$\begin{cases} L_{s,i} = f(L_{s+1,i}, L_{s+1,i+2^s}), \\ L_{s,i+2^s} = g(L_{s+1,i}, L_{s+1,i+2^s}, \hat{v}_{s,i}). \end{cases} \tag{2}$$

where $L_{s,i}$ and $\hat{v}_{s,i}$ are the soft LLR value and the hard-bit estimation at the s-th stage and the i-th bit, respectively, and the min-sum approximation formulations of the functions f and g in (2) are

$$\begin{cases} f(a, b) = \min(|a|, |b|)sgn(a)sgn(b), \\ g(a, b, c) = b + (1 - 2c)a. \end{cases} \tag{3}$$

This min-sum formulation has two benefits: first, it allows for efficient hardware implementation [18]; and second, it

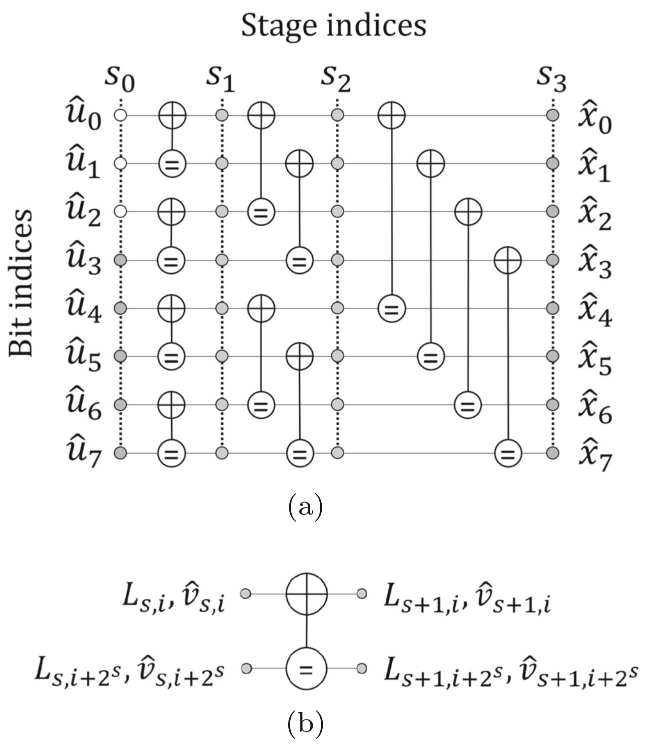
Stage indices
s_0
s_1
s_2
s_3
Bit indices
$\hat{u}_0$ $\hat{u}_1$ $\hat{u}_2$ $\hat{u}_3$ $\hat{u}_4$ $\hat{u}_5$ $\hat{u}_6$ $\hat{u}_7$
$\hat{x}_0$ $\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$ $\hat{x}_7$
(a)
$L_{s,i}, \hat{v}_{s,i}$
$L_{s+1,i}, \hat{v}_{s+1,i}$
$L_{s,i+2^s}, \hat{v}_{s,i+2^s}$
$L_{s+1,i+2^s}, \hat{v}_{s+1,i+2^s}$
(b)

Figure 1 **a** SC decoding on the factor graph of $\mathcal{P}(8,5)$ with $\{u_0, u_1, u_2\} \in \mathcal{A}^c$, **b** a PE.

allows the decoder to initialize the LLR values directly by the channel outputs without the need to estimate the channel noise power σ^2 as is required in (1) [21, Section 5.5.1]. Thus in this paper we use

$$\boldsymbol{L}_n = \boldsymbol{y}. \tag{4}$$

The hard-bit values of the PE are computed as

$$\begin{cases} \hat{v}_{s+1,i} = \hat{v}_{s,i} \oplus \hat{v}_{s,i+2^s} \\ \hat{v}_{s+1,i+2^s} = \hat{v}_{s,i+2^s}. \end{cases} \tag{5}$$

The soft LLR values at the n-th stage are initialized to $\boldsymbol{L}_n$ and the hard-bit estimation at the 0-th stage is obtained as

$$\hat{u}_i = \hat{v}_{0,i} = \begin{cases} 0 & \text{if } u_i \in \mathcal{A}^c, \\ \frac{1-sgn(L_{0,i})}{2} & \text{otherwise.} \end{cases} \tag{6}$$

2.3 Dynamic Successive Cancellation Flip Decoding

The error-correction performance of SC decoding for short to moderate block lengths is not satisfactory. To improve its error-correction performance, a CRC of length c is concatenated to the message word of polar codes to check whether SC decoding succeeded or not. If the estimated message word $\hat{\boldsymbol{u}}$ does not satisfy the CRC after the initial SC decoding attempt, a secondary SC decoding attempt is made by flipping the estimation of an information bit in $\hat{\boldsymbol{u}}$ that is most likely to be erroneous. This process can be performed multiple times by applying a predetermined number of SC decoding attempts, with each attempt flipping the estimation of a different information bit. If the resulting message word after one of the SC decoding attempts satisfies the CRC, the decoding is declared successful. This algorithm is referred to as SCF decoding [2]. The main problem associated with SCF decoding is that only the first erroneous bit after the initial SC decoding can be corrected. However, it is common that even after the first erroneous bit is corrected, the resulting message word still contains erroneous bits. Therefore, further flipping attempts for the additional erroneous bits are required. DSCF decoding was introduced in [5] to address this problem.

Let $\mathcal{E}_\omega = \{i_1, \ldots, i_\omega\}$, where $\{u_{i_1}, \ldots, u_{i_\omega}\} \subset \mathcal{A}$, be the set of bit-flipping positions of order ω such that $i_1 < \cdots < i_\omega$, $0 \leq \omega \leq K + c$, and $|\mathcal{E}_\omega| = \omega$. Note that $\mathcal{E}_0 = \emptyset$. In the course of DSCF decoding, the hard-bit estimations of all the bit indices in $\mathcal{E}_\omega$ are flipped. The set $\mathcal{E}_\omega$ is constructed progressively based on the set $\mathcal{E}_{\omega-1} = \{i_1, \ldots, i_{\omega-1}\}$. In fact, if SC decoding fails after flipping all the bit-flipping positions in $\mathcal{E}_{\omega-1}$, i_ω is added to $\mathcal{E}_{\omega-1}$ to form $\mathcal{E}_\omega$ and an additional SC decoding attempt is performed by flipping the bit estimation at all the bit-flipping positions in $\mathcal{E}_\omega$. Furthermore, a maximum number of decoding attempts m_ω is imposed on the decoder to limit the computational complexity in practice. The bit-flipping process of the ω-th error order under SC decoding can be written as

$$\hat{u}[\mathcal{E}_\omega]_i = \begin{cases} 0 & \text{if } u_i \in \mathcal{A}^c, \\ \frac{1+sgn(L[\mathcal{E}_\omega]_{0,i})}{2} & \text{if } u_i \in \mathcal{A}, i \in \mathcal{E}_\omega, \\ \frac{1-sgn(L[\mathcal{E}_\omega]_{0,i})}{2} & \text{otherwise,} \end{cases} \tag{7}$$

where $\boldsymbol{L}[\mathcal{E}_\omega]$ is the vector of LLR values obtained at the ω-th error order.

Let

$$p_i^*(\mathcal{E}_{\omega-1}) = \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_i = u_i | \boldsymbol{y}, \hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]_0^{i-1} = \boldsymbol{u}_0^{i-1}), \tag{8}$$

where

$$\begin{cases} \hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]_0^{i-1} & = \{\hat{u}[\mathcal{E}_{\omega-1}]_0, \hat{u}[\mathcal{E}_{\omega-1}]_1, \ldots, \hat{u}[\mathcal{E}_{\omega-1}]_{i-1}\}, \\ \boldsymbol{u}_0^{i-1} & = \{u_0, u_1, \ldots, u_{i-1}\}. \end{cases}$$

The probability that SC decoding is successful after flipping all the bit-flipping positions in $\mathcal{E}_\omega$ is then defined as [5]

$$P_{i_\omega} = \prod_{\substack{\forall i \in \mathcal{A} \setminus \mathcal{E}_\omega \\ i < i_\omega}} p_i^*(\mathcal{E}_{\omega-1}) \times \prod_{\forall i \in \mathcal{E}_\omega} \left(1 - p_i^*(\mathcal{E}_{\omega-1})\right). \tag{9}$$

Therefore, the bit-flipping position i_ω^* that maximizes the probability of $\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]$ being correctly decoded is

$$i_\omega^* = \underset{\substack{\forall i_\omega \in \mathcal{A}, i_{\omega-1} < i_\omega \leq N-1 \\ \mathcal{E}_\omega = \mathcal{E}_{\omega-1} \cup i_\omega}}{\arg\max} P_{i_\omega}. \tag{10}$$

Note that the probability $p_i^*(\mathcal{E}_{\omega-1})$ cannot be obtained during the course of decoding as the values of the elements of $\boldsymbol{u}$ are unknown to the decoder [5]. As a result, DSCF decoding uses a known probability $p_i(\mathcal{E}_{\omega-1})$ to estimate $p_i^*(\mathcal{E}_{\omega-1})$. The known probability $p_i(\mathcal{E}_{\omega-1})$ is defined as

$$\begin{aligned} p_i(\mathcal{E}_{\omega-1}) &= \max\Big(\Pr(\hat{u}[\mathcal{E}_{\omega-1}]_i = 0|\boldsymbol{y}, \hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]_0^{i-1}), \\ &\qquad \Pr\Big(\hat{u}[\mathcal{E}_{\omega-1}]_i = 1|\boldsymbol{y}, \hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]_0^{i-1}\Big)\Big) \\ &= \frac{1}{1+\exp\left(-|L[\mathcal{E}_{\omega-1}]_{0,i}|\right)}, \end{aligned} \tag{11}$$

where $L[\mathcal{E}_{\omega-1}]_{0,i}$ is the corresponding LLR value of $\hat{u}[\mathcal{E}_{\omega-1}]_i$. It was shown in [5] that the estimation in (11) is not accurate. Therefore, [5] introduced a perturbation parameter α to have a better estimation of $p_i^*(\mathcal{E}_{\omega-1})$ as

$$p_i^*(\mathcal{E}_{\omega-1}) \approx \frac{1}{1+\exp\left(-\alpha|L[\mathcal{E}_{\omega-1}]_{0,i}|\right)}. \tag{12}$$

It should be noted that $\alpha \in \mathbb{R}^+$ is a scaling factor for the magnitude of the LLR values and is determined by a Monte-Carlo simulation. To enable a trade-off between decoding latency and error-correction performance, instead of only flipping the most probable bit-flipping position, DSCF decoding attempts to improve SC decoding with a list of most probable bit-flipping indices i_ω^* at each error order ω [5].

In order to have numerically stable computations in the hardware implementation of the DSCF decoder, the bit-flipping metric in (9) can be written in the log-likelihood (LL) domain as [5]

$$\begin{aligned} Q_{i_\omega} &= -\frac{1}{\alpha}\ln(P_{i_\omega}) \\ &= \sum_{\substack{\forall i \in \mathcal{A} \\ i \le i_\omega}} \frac{1}{\alpha}\ln\left(1+\exp\left(-\alpha|L[\mathcal{E}_{\omega-1}]_{0,i}|\right)\right) \\ &\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|. \end{aligned} \tag{13}$$

Consequently, the most probable bit-flipping position i_ω^* can be found in the LL domain as

$$i_\omega^* = \mathop{\arg\min}_{\substack{\forall i_\omega \in \mathcal{A}, i_{\omega-1} < i_\omega \le N-1 \\ \mathcal{E}_\omega = \mathcal{E}_{\omega-1} \cup i_\omega}} Q_{i_\omega}. \tag{14}$$

3 Neural Successive Cancellation Flip Decoding

In this section, a novel low-complexity bit-flipping metric computation scheme is presented to allow efficient hardware implementation. Then, a machine learning framework is introduced to optimize the parameter in the proposed bit-flipping metric computation scheme.

3.1 Bit-flipping Metric Computation

Efficient hardware implementation of DSCF decoding is contingent on the efficient implementation of the bit-flipping metric in (13). However, (13) involves logarithmic and exponential functions that are not hardware friendly. A common approach to approximate the logarithmic and exponential function in (13) is to use the rectifier linear unit (ReLU) as [4]

$$\ln(1+\exp(x)) \approx ReLU(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

However, since $\alpha > 0$, $-\alpha|L[\mathcal{E}_{\omega-1}]_{0,i}| < 0$. Therefore, (13) can be simplified as

$$\begin{aligned} Q_{i_\omega} &\approx \sum_{\substack{\forall i \in \mathcal{A} \\ i \le i_\omega}} \frac{1}{\alpha} ReLU\left(-\alpha|L[\mathcal{E}_{\omega-1}]_{0,i}|\right) \\ &\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}| \\ &= \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|, \end{aligned} \tag{16}$$

which is independent of the perturbation parameter α. Figure 2 shows the effect of the simplification in (16) on the error-correction performance of DSCF decoding in terms of FER for $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. The polar codes are concatenated with a 24-bit CRC used in the control channel of 5G standard [1]. In this figure, the FER curve of the DSCF decoder at error order ω with m_ω decoding attempts is denoted as DSCF-(ω, m_ω). The value of α is set to 0.3 as it provides a good result across a wide range of SNR values [5].[1] The FER of the ideal DSCF decoder, denoted as I-DSCF, where the erroneous bits up to the ω-th error order are always accurately corrected, is also plotted for comparison. As seen from Figure 2, the over-simplification of the bit-flipping metric calculation in (16) results in 0.15, 0.3, and 0.45 dB error-correction performance loss for $\mathcal{P}(512, 256)$ compared to the DSCF decoder when $\omega = \{1, 2, 3\}$ and $m_\omega = \{10, 100, 400\}$, respectively, at a target FER of 10^{-4}. For $\mathcal{P}(512, 384)$, the corresponding FER degradation caused by the over-simplification operations is 0.05, 0.16, and 0.23 dB for $\omega = \{1, 2, 3\}$ and $m_\omega = \{10, 100, 400\}$ at the same target FER of 10^{-4}, respectively.

To address this issue, we propose to use a perturbation parameter $\beta \in \mathbb{R}^+$ that unlike α, is an additive positive parameter, and like α, tries to improve the estimation of $p_i^*(\mathcal{E}_{\omega-1})$. We write the proposed estimation of $p_i^*(\mathcal{E}_{\omega-1})$ as

$$p_i^*(\mathcal{E}_{\omega-1}) \approx \frac{1}{1+\exp\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right)}. \tag{17}$$

[1]Since the channel output y is directly used in this paper as the decoding input, we set $\alpha = \frac{0.6}{\sigma^2}$ to obtain the same FER performance of the DSCF decoder in [5].

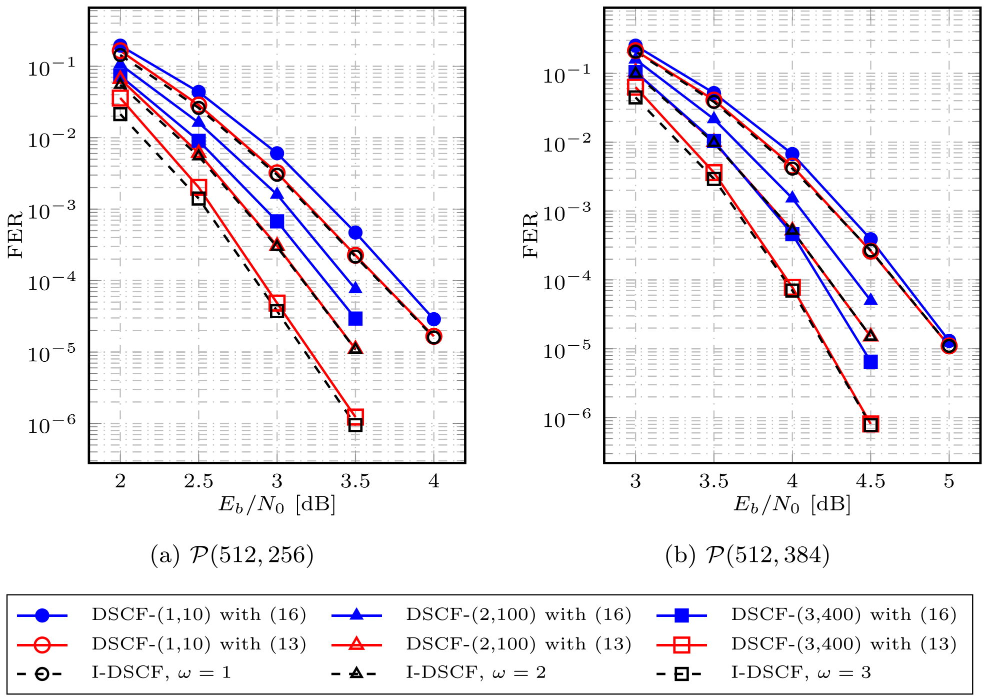

Figure 2 Effect of the simplification in (16) on the FER of DSCF decoding for $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. The polar codes are concatenated with a 24-bit CRC used in 5G standard. The ideal DSCF decoder (I-DSCF) is also plotted as a reference.

In addition, we propose to use a bit-flipping metric in the LL domain, Q_{i_ω}, which is tailored to the proposed $p_i^*(\mathcal{E}_{\omega-1})$ in (17) as

$$\begin{aligned} Q_{i_\omega} &= -\ln(P_{i_\omega}) + \omega\beta \\ &= \sum_{\substack{\forall i \in \mathcal{A} \\ i \le i_\omega}} \ln\left(1 + \exp\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right)\right) \\ &\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|, \end{aligned} \tag{18}$$

where we used the fact that $\omega\beta$ is a constant and it will not affect the selection of i_ω^* in (18).

Let us now use the ReLU function in (15) to simplify the proposed bit-flipping metric in (18) as

$$\begin{aligned} Q_{i_\omega} &\approx \sum_{\substack{\forall i \in \mathcal{A} \\ i \le i_\omega}} ReLU\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right) \\ &\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|, \end{aligned} \tag{19}$$

where we used the fact that if $\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}| > 0$, then

$$ReLU\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right) = \beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|.$$

It can be seen that the resulting bit-flipping metric is dependent on the value of β, and it is hardware friendly since only additions are required for the metric computation. Note that in this paper, a different value of β is used to calculate the bit-flipping metric in (19) at each error order.

Algorithm 1: NSCF Decoding Algorithm.

```
Input  : $\boldsymbol{y}, \boldsymbol{m} = \{m_1, m_2, \ldots, m_\omega\}, \beta$
Output: $\hat{\boldsymbol{u}}$
// Perform NSCF decoding upto the
   $\omega$-th error order
1  for $\tau \leftarrow 0$ to $\omega$ do
       // Initialize the bit-flipping
          data structure
2      if $\tau \leftarrow 0$ then
3          $S_0 \leftarrow [\emptyset, \mathcal{E}_0]$
4      $S_{\tau+1} \leftarrow \emptyset$
       // SC decoding with bit-flipping
          operations
5      forall the $\mathcal{E}_\tau$ in $S_\tau$ do
6          Perform SC decoding based on (2)-(5) and (7)
           to obtain $\hat{u}[\mathcal{E}_\tau]_i$
7          if $i > \arg\max_{\forall j \in \mathcal{E}_\tau}\{\mathcal{E}_\tau\}$ and $i \in \mathcal{A}$ then
8              Form the candidate bit-flipping set
               $\mathcal{E}_{\tau+1} \leftarrow \mathcal{E}_\tau \cup i$
9              Obtain $Q_{i_{\tau+1}}$ using (19)
10             InsertionSort $\left(S_{\tau+1}, [Q_{i_{\tau+1}}, \mathcal{E}_{\tau+1}]\right)$
11         if $\hat{\boldsymbol{u}}[\mathcal{E}_\tau]$ satisfies CRC then
12             $\hat{\boldsymbol{u}} = \hat{\boldsymbol{u}}[\mathcal{E}_\tau]$
13             return $\hat{\boldsymbol{u}}$
```

We summarize the proposed NSCF decoding algorithm which corrects up to the ω-th error order under SC decoding in Algorithm 1. We denote by $S_\tau (0 \le \tau \le \omega)$ a data structure whose elements contain a pair of $[Q_{i_\tau}, \mathcal{E}_\tau]$, where Q_{i_τ} is the bit-flipping metric associated with a bit-flipping set $\mathcal{E}_\tau$ at the τ-th error order. Note that

$$|S_\tau|_{\max} = \begin{cases} 0 & \text{if } \tau = 0 \\ m_\tau - \sum_{k=0}^{\tau-1} m_k & \text{otherwise,} \end{cases}$$

thus the maximum number of all the additional decoding attempts up to the τ-th error order is m_τ.

At the τ-th error order, the proposed decoder loops over all the candidate bit-flipping sets $\mathcal{E}_\tau$ stored in S_τ. SC decoding with bit-flipping operations is then carried out given a bit-flipping set $\mathcal{E}_\tau$. At the i-th information bit and if $i > \arg\max_{\forall j \in \mathcal{E}_\tau}\{\mathcal{E}_\tau\}$, a new candidate bit-flipping set is constructed for the $(\tau+1)$-th error order by concatenating the current information bit position to the current bit-flipping set $\mathcal{E}_\tau$, i.e., $\mathcal{E}_{\tau+1} \leftarrow \mathcal{E}_\tau \cup i$. The bit-flipping metric $Q_{i_{\tau+1}}$, which is associated with the newly constructed bit-flipping set $\mathcal{E}_{\tau+1}$, is then calculated using the proposed bit-flipping metric computation in (19). The data structure $S_{\tau+1}$ is then updated with the newly constructed element $[Q_{i_{\tau+1}}, \mathcal{E}_{\tau+1}]$ by performing an insertion sort using the new bit-flipping metric $Q_{i_{\tau+1}}$. If $Q_{i_{\tau+1}}$ is among the $|S_{\tau+1}|_{\max}$ smallest bit-flipping metrics of $S_{\tau+1}$, the new element is inserted in $S_{\tau+1}$, and the element that has the largest bit-flipping metric is discarded. Otherwise, the newly constructed element is discarded. This process is carried out in the $InsertionSort(\cdot)$ function denoted in Algorithm 1. Note that if the resulting estimated message word given the bit-flipping set $\mathcal{E}_\tau$, i.e. $\hat{\boldsymbol{u}}[\mathcal{E}_\tau]$, satisfies the CRC verification, the decoding terminates and outputs $\hat{\boldsymbol{u}}[\mathcal{E}_\tau]$ as the estimated message word. Also note that the proposed NSCF decoder reverts to the conventional DSCF decoder by replacing (19) in Algorithm 1 with (13).

3.2 Parameter Optimization

In this paper, we formalize the optimization problem of the additive parameter β as a separate classification problem at each error order and use ML techniques to train β. As observed from (19), the bit-flipping metric computation takes the absolute values of the soft messages given by SC decoding as the input. Therefore, the bit-flipping metric computation does not depend on the value of u_i. Thus, it allows the use of the all-zero codeword for the training of β, which simplifies the data collection process, as also observed in [8, 19].

In [7], the decoding process is modeled as a deep neural network that consists of ω unfolded DSCF decoding attempts. The training data used to train the parameter at the ω-th error order in [7] includes both the samples that cannot be correctly decoded and those that can be correctly decoded up to the $(\omega-1)$-th error order. As a result, the size of the training dataset is excessively large. For example, the framework introduced in [7] requires 2.5×10^5 samples for $\omega = 2$. Unlike [7], in this paper we consider the parameter optimization of each error order individually. To train the parameter at the ω-th error order, only the frames that do not satisfy the CRC verification of the ideal DSCF decoder at the $(\omega-1)$-th error order are used. In fact, the frames that are not decoded correctly contribute to the training and optimization of parameter β. It is observed that with the new training process, only 5000 samples are required to train β at each error order.

Let $T_{\omega-1}$ be the set of the bit-flipping indices, and $t_{\omega-1}$ be the $(\omega-1)$-th bit-flipping index of the $(\omega-1)$-th ideal DSCF decoder. In addition, let $\boldsymbol{L}[T_{\omega-1}]$ be the LLR values of the $(w-1)$-th ideal DSCF decoder given that the corresponding hard decisions of $\boldsymbol{L}[T_{\omega-1}]_0$ do not satisfy the CRC verification. For the rest of this paper, since we only consider non-frozen bits, all the bit indices only indicate non-frozen bit positions. Thus, they are in the range of $[0, K+c-1]$.

The bit-flipping metric rendered for the i_ω-th bit index, $t_{\omega-1} < i_\omega < K+c$, of the ideal DSCF decoder is written as

$$Q_{i_\omega} \approx \sum_{0 \le i \le i_\omega} ReLU\left(\beta - |L[T_{\omega-1}]_{0,i}|\right) + \sum_{\forall i \in \{T_{\omega-1} \cup i_\omega\}} |L[T_{\omega-1}]_{0,i}|. \tag{20}$$

The value of Q_{i_ω} is normalized using the soft-min function $\delta(\cdot)$ as

$$\tilde{O}_{i_\omega} = \delta(Q_{i_\omega}) = \frac{\exp(-Q_{i_\omega})}{\sum_{j=t_{\omega-1}+1}^{K+C-1} \exp(-Q_{j_\omega})}. \tag{21}$$

It can be seen that for all values of i_ω, $0 < \tilde{O}_{i_\omega} < 1$, and

$$i_\omega^* = \underset{\substack{\forall i_\omega \\ t_{\omega-1} < i_\omega < K+c}}{\arg\min}\ Q_{i_\omega} = \underset{\substack{\forall i_\omega \\ t_{\omega-1} < i_\omega < K+c}}{\arg\max}\ \tilde{O}_{i_\omega}, \tag{22}$$

where i_ω^* is the most probable bit-flipping position. Note that $\tilde{O}_{i_\omega}$ can be viewed as the predicted bit-flipping probability. Let us define the training label O_{i_ω} as

$$O_{i_\omega} = \begin{cases} 1 & \text{if } i_\omega = t_\omega, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

In this paper, the binary cross-entropy loss function is used to quantify the differences of the estimated value $\tilde{O}_{i_\omega}$ and the exact training label O_{i_ω}. This can be written as

$$\mathcal{L} = -\sum_{i_\omega = t_{\omega-1}+1}^{K+C-1} \left[O_{i_\omega} \ln \tilde{O}_{i_\omega} + (1 - O_{i_\omega}) \ln(1 - \tilde{O}_{i_\omega}) \right]. \tag{24}$$

By using the stochastic gradient-descent optimization technique or its variants, the parameter β can be optimized to minimize the loss $\mathcal{L}$ [17]. The gradient of the objective loss function with respect to the additive parameter β can be obtained as

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i_\omega = t_{\omega-1}+1}^{K+C-1} \frac{\partial \mathcal{L}}{\partial \tilde{O}_{i_\omega}} \frac{\partial \tilde{O}_{i_\omega}}{\partial Q_{i_\omega}} \frac{\partial Q_{i_\omega}}{\partial \beta} \tag{25}$$

where

$$\frac{\partial \mathcal{L}}{\partial \tilde{O}_{i_\omega}} = \frac{\tilde{O}_{i_\omega} - O_{i_\omega}}{\tilde{O}_{i_\omega}(1 - \tilde{O}_{i_\omega})}, \tag{26}$$

$$\frac{\partial \tilde{O}_{i_\omega}}{\partial Q_{i_\omega}} = \tilde{O}_{i_\omega}(\tilde{O}_{i_\omega} - 1), \tag{27}$$

$$\frac{\partial Q_{i_\omega}}{\partial \beta} = \sum_{j=0}^{i_\omega} \mathbb{1}_{\beta > |L[T_{\omega-1}]_{0,j}|}, \tag{28}$$

and $\mathbb{1}_{a>b}$ $(a, b \in \mathbb{R})$ is an indicator function such that

$$\mathbb{1}_{a>b} = \begin{cases} 1 & \text{if } a > b, \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Substituting (26)-(28) into (25) gives

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i_\omega = t_{\omega-1}+1}^{K+C-1} \left(O_{i_\omega} - \tilde{O}_{i_\omega}\right) \sum_{j=0}^{i_\omega} \mathbb{1}_{\beta > |L[T_{\omega-1}]_{0,j}|}. \tag{30}$$

In this paper, we use Root Mean Square Propagation (RMSProp), a variant of the SGD optimization technique, to update the parameter β [15]. The additive parameter β is then updated as [15]

$$\beta := \beta - \frac{\lambda}{\sqrt{\mu_\beta}} \frac{\partial \mathcal{L}}{\partial \beta}, \tag{31}$$

where μ_β is a running average of the magnitudes of recent gradients for β that is defined as [15]

$$\mu_\beta := \gamma \mu_\beta + (1-\gamma) \left(\frac{\partial \mathcal{L}}{\partial \beta}\right)^2, \tag{32}$$

and λ and γ are the learning rate and the forgetting factor, respectively. The value of μ_β is initialized for the first update as

$$\mu_\beta = (1-\gamma) \left(\frac{\partial \mathcal{L}}{\partial \beta}\right)^2. \tag{33}$$

It is worth mentioning that by manually deriving $\frac{\partial \mathcal{L}}{\partial \beta}$ as denoted in (30), the additive parameter β can be optimized using a SGD-based optimization technique at the decoder side without the need of a sophisticated machine learning library, which paves the way for a dedicated hardware implementation that optimizes β online using the all-zero codeword pilot signals.

3.3 Quantization Scheme

If training with quantization is considered, the additive parameter β is optimized by taking into account the quantization effect caused by the SC decoding operations. At a given error order ω, quantization operations are first applied to the ideal DSCF decoders to obtain the quantized values of $\boldsymbol{L}[T_{\omega-1}]$. In addition, the forward pass computations of the bit-flipping metric in (20) are also quantized. On the other hand, all other computations in (21)-(33) required for the backward pass to obtain the partial derivative of β are carried out in the full-precision representation. After each parameter update in (31), the value of β is quantized for the next forward pass computation carried out in (20). Note that this technique is widely used for the training of quantized deep neural networks [12]. Given the quantized value of β, in the decoding phase, NSCF decoding performs all of the SC decoding operations as well as the bit-flipping metric computations in (19) using a single quantization scheme, which is characterized by a set of quantization parameters.

To find the quantization parameters for the proposed NSCF decoder, we evaluate the quantization parameters on the ideal DSCF decoder and use those quantization parameters in the proposed NSCF decoder. Let $q(n, m)$ denote a quantization configuration where n and m indicate the number of binary bits used to represent the integral and fractional parts of a floating-point number, respectively. Figure 3 compares the FER of the ideal DSCF decoder when the soft messages used by SC decoding are quantized using $q(2, 3)$, $q(3, 2)$, and $q(3, 3)$ formats. As observed from Figure 3, the $q(3, 3)$ format introduces almost no FER performance degradation compared to the full-precision ideal DSCF decoder for both $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. Therefore, we select $q(3, 3)$ as the quantization scheme for the proposed NSCF decoder.

4 Evaluation

In this section, we first provide the training results for both full-precision and quantized scenarios. We then evaluate the error-correction performance and decoding latency of the proposed decoders.

4.1 Parameter Optimization Results

In this paper, we use Pytorch [20] to implement the parameter optimization framework[2]. We use 5000 samples

[2]We manually implement the computations in (20)-(33) instead of using the built-in automatic differentiation mechanism and SGD-based optimizers supported by Pytorch. The main purpose of using Pytorch is to make use of its GPU support to reduce the training time.

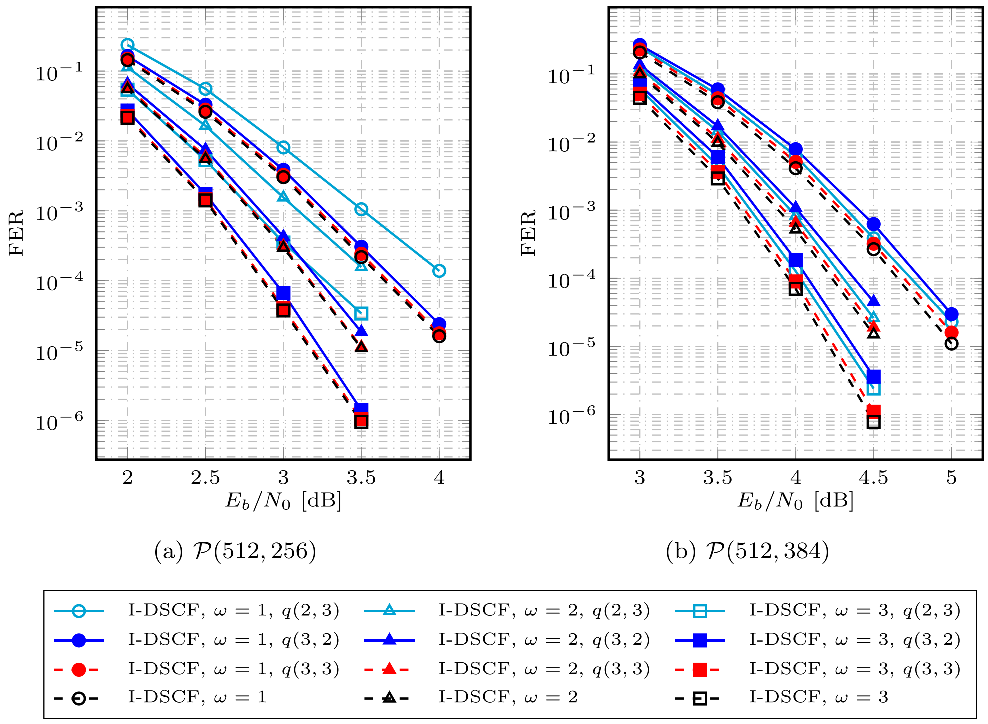

Figure 3 Effect of quantization on the FER of ideal DSCF decoding for $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. The polar codes are concatenated with a 24-bit CRC.

to optimize β at each error order $\omega \in \{1, 2, 3\}$ individually, with 4000 samples used for training and 1000 samples used for validation. In addition, the training samples are obtained at $E_b/N_0 = 3.0$ dB and $E_b/N_0 = 4.0$ dB for $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$, respectively. For both full-precision and quantized scenarios, the learning rate λ and the forgetting factor γ used in (31) and (32) are set to 5×10^{-4} and 0.9, respectively. The mini-batch size is 200 and the number of training epochs is 40. Initially, the value of β at each error order is drawn from an i.i.d distribution in the range of $(0, 5)$. When training with quantization, β is in $q(2, 3)$ format and the bit-flipping metric Q_{i_ω} is in $q(3, 3)$ format. We do not use a sign bit for the quantized values of β and Q_{i_ω}.

Figure 4 illustrates the training (validation) loss and accuracy when the β parameter is optimized for $\omega = 3$ with $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. As the optimization of β is formalized as a classification process, the training (validation) accuracy depicted in Figure 4 indicates the probability that the estimated error bit i^*_ω is the correct bit-flipping index t_ω. In the ML literature, the accuracy considered in Figure 4 is also referred as the top-1 accuracy in a classification task [16]. It can be observed that for both full-precision and quantized scenarios, the training loss and the validation loss values are almost similar, indicating that the β parameter is generalized well for unseen samples. The value of β is then selected at the training epoch that has the highest validation accuracy. It can also be observed that the full-precision model provides a smoother learning curves compared to those of the quantized model. This is because the quantized model often requires more training epochs than the full-precision model to update the parameter.

Table 1 provides the optimized values of β for both full-precision and quantized formats for $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. As the values of β at all error orders are within the range of $(0, 1)$, during the decoding phase, β is quantized using the $q(0, 3)$ format.

4.2 Error-Correction Performance

The error-correction performance of the proposed decoders in terms of FER are evaluated using the optimized values of β. We use the same polar codes $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$ as in Figures 2 and 3 to evaluate the error-correction performance of the proposed decoders. The FERs of the DSCF and NSCF at error order ω, with a maximum of m_ω attempts, are denoted as DSCF-(ω, m_ω) and NSCF-(ω, m_ω), respectively, and are shown in Figures 5 and 6. In addition, the FERs of the ideal DSCF decoder, denoted as I-DSCF, are plotted for comparison. In this paper, we set $\omega \in \{1, 2, 3\}$ and the number of maximum decoding attempts for all the DSCF and NSCF decoders are $m_\omega \in \{10, 100, 400\}$, respectively. The bit-flipping metric used for DSCF decoding for all the simulations is calculated as in (13).

It can be seen in Figure 5 that for $\omega = \{1, 2\}$, the proposed decoder in full-precision and in $q(3, 3)$ formats experiences almost no error-correction performance degradation compared to the ideal DSCF decoder. At $\omega = 3$, the error-correction performance of the NSCF decoder in

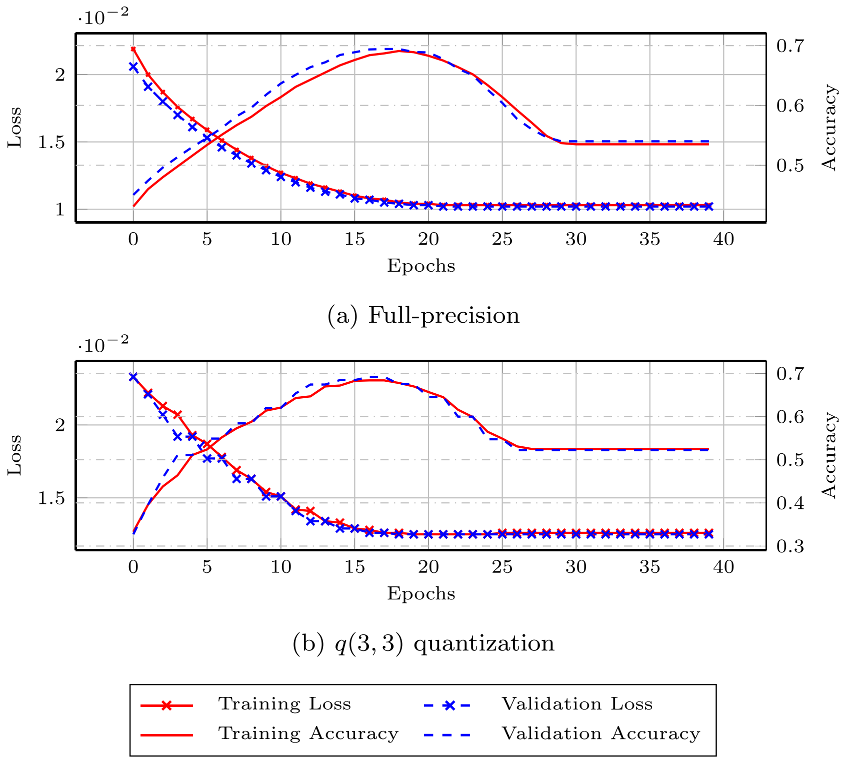

Figure 4 Plot of training (validation) accuracy and loss of the full-precision and quantized models when $\omega = 3$ for $\mathcal{P}(512, 256)$. The value of β is selected at the epoch that has the highest validation accuracy.

full-precision and in quantized schemes only has a degradation of less than 0.1 dB at the target FER of 10^{-4} when compared to that of the ideal DSCF decoder for both $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. On the other hand, when compared with that of the DSCF decoder, the FER performance loss of the proposed decoder is negligible at all considered E_b/N_0 values.

Figure 6 shows the FER performances of the proposed NSCF decoders and those of the CRC-aided SCL (CA-SCL) decoders [4] with list size m_L, denoted as CA-SCLm_L, where $m_L \in \{2, 4, 8, 16\}$. It can be seen that for $\mathcal{P}(512, 256)$, at the target FER of 10^{-4}, compared to CA-SCL with $m_L \in \{2, 4, 8\}$, the NSCF decoders at $m_\omega \in \{1, 2, 3\}$ obtain the FER performance gains of up to 0.1 dB. Moreover, for $\mathcal{P}(512, 256)$, the proposed NSCF decoder at $m_\omega = 3$ only experiences an error-correction performance loss of less than 0.1 dB compared to CA-SCL16, at the same target FER. In the case of $\mathcal{P}(512, 384)$, at the target FER of 10^{-4} the NSCF decoder at $m_\omega \in \{1, 2, 3\}$ obtains the FER gains of at least 0.2 dB when compared with the CA-SCL decoder with list size $m_L \in \{2, 4, 8\}$, respectively. In addition, for $\mathcal{P}(512, 384)$ the NSCF decoder at $\omega = 3$ has a slightly better error correction performance when compared with that of CA-SCL16 at the same target FER.

Table 1 Optimized parameter β at each error order of the proposed NSCF decoders.

ω			1	2	3
β	$\mathcal{P}(512, 256)$	Full-precision	0.9772	0.8166	0.7046
		$q(0, 3)$	0.875	0.75	0.625
	$\mathcal{P}(512, 384)$	Full-precision	0.7993	0.3671	0.3243
		$q(0, 3)$	0.5	0.375	0.375

4.3 Complexity Reduction and Decoding Latency

Since DSCF and NSCF decoding algorithms both rely on SC decoding algorithm, the only difference in terms of computational complexity comes from the bit-flipping metric computation. Table 2 shows the average number of computations performed at $E_b/N_0 = 3$ dB for $\mathcal{P}(512, 256)$ and $E_b/N_0 = 4$ dB for $\mathcal{P}(512, 384)$, which are required by the bit-flipping metric calculation of the decoders in Figure 5. Note that the bit-flipping metric computations of the DSCF decoder and that of the proposed NSCF decoder are specified in (13) and (19), respectively.

It can be seen in Table 2 that for both full-precision and quantized schemes, the proposed NSCF decoder requires around 31% fewer total number of additions compared to the DSCF decoder at all error orders. In addition, for $\omega = \{2, 3\}$, the prediction of the quantized bit-flipping model of NSCF is less accurate compared with that of the full-precision model, thus it results in a slightly larger number of additions compared to the full-precision model. On the other hand, at $\omega = 1$, the average number additions

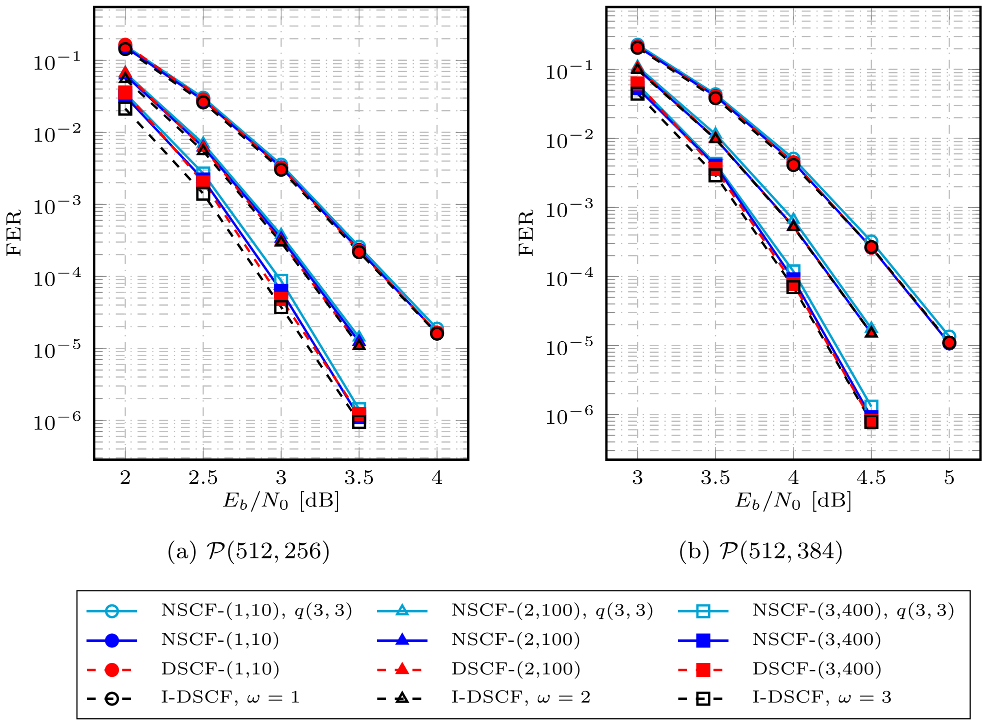

Figure 5 FER performance of the proposed decoders for $\mathcal{P}(512, 256)$ and $\mathcal{P}(512, 384)$. The polar codes are concatenated with a 24-bit CRC. The FERs of the full-precision DSCF and ideal DSCF (I-DSCF) decoders are also plotted for comparison.

required by the bit-flipping metric computation of the NSCF decoder is the same for both quantized and full-precision schemes. It can also be observed that the proposed bit-flipping metric computation completely removes the need to perform multiplications and costly transcendental computations, while only experiencing negligible error-correction performance loss when compared to DSCF as observed in Figure 5.

Figure 7 depicts the average number of decoding attempts for the DSCF decoder and the proposed NSCF decoder. It can be seen that when $E_b/N_0 > 2.5$ dB for $\mathcal{P}(512, 256)$ and $E_b/N_0 > 3.5$ dB for $\mathcal{P}(512, 384)$,

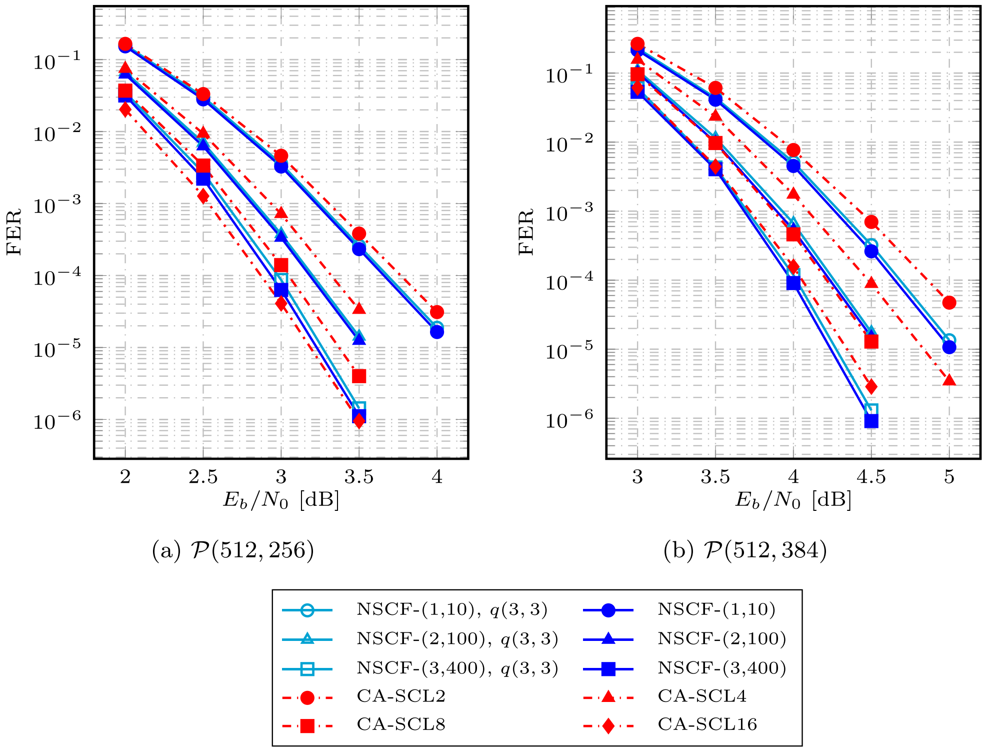

Figure 6 FER comparison of the proposed NSCF decoders and CA-SCL decoders in [4].

Table 2 Computational complexity of the bit-flipping metric in terms of the average number of operations performed for different polar codes, which are concatenated with a 24-bit CRC used in 5G.

	ω	m_ω	Decoder	ln / exp	$\times$	$+$
$\mathcal{P}(512, 256)$	1	10	DSCF	560	560	840
			NSCF	0	0	560
			NSCF-$q(3, 3)$	0	0	560
	2	100	DSCF	620.77	620.77	923.36
			NSCF	0	0	621.55
			NSCF-$q(3, 3)$	0	0	626.59
	3	400	DSCF	663.17	663.17	981.05
			NSCF	0	0	666.55
			NSCF-$q(3, 3)$	0	0	677.68
$\mathcal{P}(512, 384)$	1	10	DSCF	816	816	1224
			NSCF	0	0	816
			NSCF-$q(3, 3)$	0	0	816
	2	100	DSCF	933.67	933.67	1390.8
			NSCF	0	0	949.3
			NSCF-$q(3, 3)$	0	0	956.8
	3	400	DSCF	1020.32	1020.32	1512.7
			NSCF	0	0	1058.8
			NSCF-$q(3, 3)$	0	0	1059.9

the average number of decoding attempts of the proposed NSCF decoder is similar to that of the DSCF decoder, under the same decoding configurations. Note that the average number of decoding attempts of all the decoders depicted in Figure 7 approaches 1 at high E_b/N_0 values. This also indicates that at high SNR regime, the average complexity of all the DSCF-based decoders approaches the complexity of a single SC decoder, while their error-correction performance is comparable to that of CA-SCL decoder as observed from Figures 5 and 6.

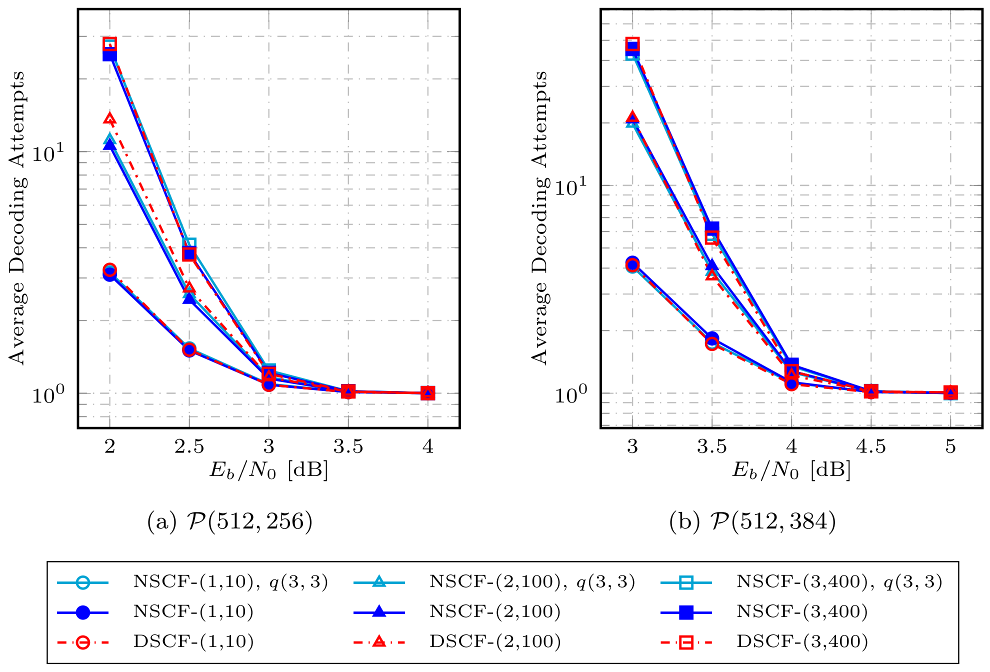

Figure 7 Average number of decoding attempts.

5 Conclusion

In this paper, we proposed a neural successive cancellation flip (NSCF) decoding algorithm for polar codes. The proposed decoder uses an additive parameter to improve the accuracy of the bit-flipping metric and the parameter is optimized by a novel training framework. The proposed decoder has the following advantages: (i) its average decoding complexity approaches that of the successive cancellation (SC) decoding at high signal-to-noise ratio (SNR) regimes; (ii) only additions are needed during the course of decoding; (iii) negligible error-correction performance loss is incurred in comparison with the ideal dynamic successive cancellation flip (DSCF) decoder. With these advantages, the proposed decoder is a potential candidate for an efficient hardware implementation of a bit-flipping decoding algorithm for polar codes.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. 3GPP (2018). Multiplexing and channel coding (Release 10) 3GPP TS 21.101 v10.4.0. http://www.3gpp.org/ftp/Specs/2018-09/Rel-10/21_series/21101-a40.zip.
2. Afisiadis, O., Balatsoukas-Stimming, A., Burg, A. (2014). A low-complexity improved successive cancellation decoder for polar codes. In *48Th asilomar conference on signals, systems, and computers* (pp. 2116–2120), https://doi.org/10.1109/ACSSC.2014.7094848.
3. Arıkan, E. (2009). Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, *55*(7), 3051–3073. https://doi.org/10.1109/TIT.2009.2021379.
4. Balatsoukas-Stimming, A., Parizi, M.B., Burg, A. (2015). LLR-Based successive cancellation list decoding of polar codes. *IEEE Transactions on Signal Processing*, *63*(19), 5165–5179. https://doi.org/10.1109/TSP.2015.2439211.
5. Chandesris, L., Savin, V., Declercq, D. (2018). Dynamic-SCFlip decoding of polar codes. *IEEE Transactions on Communications*, *66*(6), 2333–2345. https://doi.org/10.1109/TCOMM.2018.2793887.
6. Condo, C., Ercan, F., Gross, W.J. (2018). Improved successive cancellation flip decoding of polar codes based on error distribution. In *IEEE wireless communications and networking conference workshops* (pp. 19–24), https://doi.org/10.1109/WCNCW.2018.8368991.
7. Doan, N., Hashemi, S.A., Ercan, F., Tonnellier, T., Gross, W.J. (2019). Neural dynamic successive cancellation flip decoding of polar codes. In *IEEE international workshop on signal processing systems* (pp. 272–277), https://doi.org/10.1109/SiPS47522.2019.9020513.
8. Doan, N., Hashemi, S.A., Mambou, E.N., Tonnellier, T., Gross, W.J. (2019). Neural belief propagation decoding of CRC-polar concatenated codes. In *IEEE international conference on communications* (pp. 1–6), https://doi.org/10.1109/ICC.2019.8761399.
9. Ercan, F., Condo, C., Gross, W.J. (2019). Improved bit-flipping algorithm for successive cancellation decoding of polar codes. *IEEE Transactions on Communications*, *67*(1), 61–72. https://doi.org/10.1109/TCOMM.2018.2873322.
10. Ercan, F., Condo, C., Hashemi, S.A., Gross, W.J. (2017). On error-correction performance and implementation of polar code list decoders for 5g. In *2017 55th annual allerton conference on communication, control, and computing (allerton)* (pp. 443–449), https://doi.org/10.1109/ALLERTON.2017.8262771.
11. Ercan, F., Condo, C., Hashemi, S.A., Gross, W.J. (2018). Partitioned successive-cancellation flip decoding of polar codes. In *IEEE international conference on communications* (pp. 1–6), https://doi.org/10.1109/ICC.2018.8422464.
12. Han, S., Mao, H., Dally, W.J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. International Conference on Learning Representative, arXiv:1510.00149.
13. Hashemi, S.A., Condo, C., Ercan, F., Gross, W.J. (2017). Memory-efficient polar decoders. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *7*(4), 604–615. https://doi.org/10.1109/JETCAS.2017.2764421.
14. Hashemi, S.A., Condo, C., Gross, W.J. (2017). Fast and flexible successive-cancellation list decoders for polar codes. *IEEE Transactions on Signal Processing*, *65*(21), 5756–5769. https://doi.org/10.1109/TSP.2017.2740204.
15. Hinton, G., Srivastava, N., Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
16. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).
17. LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.
18. Leroux, C., Raymond, A.J., Sarkis, G., Gross, W.J. (2013). A semi-parallel successive-cancellation decoder for polar codes. *IEEE Transactions on Signal Processing*, *61*(2), 289–299. https://doi.org/10.1109/TSP.2012.2223693.
19. Nachmani, E., Marciano, E., Lugosch, L., Gross, W.J., Burshtein, D., Be'ery, Y. (2018). Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*, *12*(1), 119–131. https://doi.org/10.1109/JSTSP.2017.2788405.
20. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A. (2017). Automatic differentiation in pytorch.
21. Ryan, W., & Lin, S. (2009). *Channel codes: classical and modern*. Cambridge: Cambridge University Press.
22. Tal, I., & Vardy, A. (2015). List decoding of polar codes. *IEEE Transactions on Information Theory*, *61*(5), 2213–2226. https://doi.org/10.1109/TIT.2015.2410251.