Neural Dynamic Successive Cancellation Flip Decoding of Polar Codes

Nghia Doan*, Seyyed Ali Hashemi[†], Furkan Ercan*, Thibaud Tonnellier*, Warren J. Gross*

*Department of Electrical and Computer Engineering, McGill University, Canada

[†]Department of Electrical Engineering, Stanford University, USA

*{nghia.doan, furkan.ercan}@mail.mcgill.ca, {thibaud.tonnellier, warren.gross}@mcgill.ca

[†]ahashemi@stanford.edu

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 a 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 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 significant error-correction performance loss. We then introduce a training parameter and propose an approximation scheme which completely removes the need to perform transcendental computations in DSCF decoding, with almost no error-correction performance degradation.**

Index Terms—**5G, polar codes, deep learning, SC Flip.**

I. 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 tends towards infinity [1]. 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 [2]. 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 [3] to improve the error-correction performance of SC decoding for short to moderate polar codes by keeping a list of candidate codewords at each decoding step. In addition, it was observed that under SCL decoding, the error probabilities are significantly enhanced when the polar code is concatenated with a cyclic redundancy check (CRC) [3]. However, the implementation complexity of SCL decoding grows as the list size increases.

SC flip (SCF) decoding algorithm was introduced in [4] which unlike SCL 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 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 [5]–[7], however they are limited with correcting a single erroneous bit in the codeword. Dynamic SCF (DSCF) decoding [8] is a generalization of SCF-based decoding which is able to correct higher-order

erroneous information bits, i.e., DSCF decoding can correct an erroneous bit which is a result of error propagation under SC decoding, given that all previous erroneous bits were correctly flipped [8].

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 [8] comparable to that of SCL decoding. However, DSCF requires costly exponential and logarithmic computations which 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 resulting decoder can have an error-correction performance close to that of the ideal DSCF decoder. Furthermore, we show that the proposed decoder does not suffer from significant error-correction performance loss if the common hardware-friendly approximation techniques are used. In addition, a novel deep learning framework which utilizes the symmetric properties of DSCF decoding is introduced as an optimization scheme for the trainable parameter. We name the proposed decoding algorithm neural DSCF (NDSCF) decoding algorithm. Simulation results show that for a 5G polar code of length 256, with 128 information bits and concatenated with a 24-bit CRC, the proposed NDSCF decoding does not incur significant error correction performance loss in comparison with the ideal DSCF decoding, while requiring no exponential or logarithmic computations.

The rest of this paper is organized as follows. Section II briefly reviews polar codes and DSCF decoding. Section III describes the proposed NDSCF decoder. Section IV provides numerical results, and finally, Section V presents concluding remarks.

II. PRELIMINARIES

A. 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
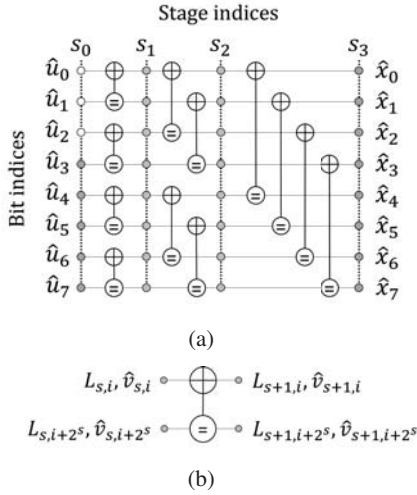
(a)

(b)

Fig. 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.

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 $\boldsymbol{y} = (\boldsymbol{1} - 2\boldsymbol{x}) + \boldsymbol{z}$, where $\boldsymbol{1}$ 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 $\boldsymbol{L}_n = \frac{2\boldsymbol{y}}{\sigma^2}$.

B. 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 Fig. 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 Fig. 1b. The computation of a PE is obtained as

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

where

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

and $L_{s,i}$ 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. The hard-bit values of the PE are computed as

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

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

$$\hat{u}_i = \hat{v}_{0,i} = \begin{cases} 0 & \text{if } u_i \in \mathcal{A}^c, \\ \frac{1-\operatorname{sgn}(L_{0,i})}{2} & \text{otherwise.} \end{cases} \quad (3)$$

C. Dynamic Successive Cancellation Flip Decoding

The error-correction performance of SC decoding for short to moderate block lengths is not satisfactory. In order 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 $\hat{\boldsymbol{u}}$ which 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 [4]. The main problem associated with SCF decoding is that only the first error bit after the initial SC decoding can be corrected. However, it is common that even after the first error bit is corrected, the resulting message word still contains error bits. Therefore, further flipping attempts for the additional error bits are required. DSCF decoding was introduced in [8] 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, where $i_1 < \cdots < i_\omega$, $0 \leq \omega \leq K + c$, 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 which can be written as

$$\hat{u}[\mathcal{E}_{\omega-1}]_i = \begin{cases} 0 & \text{if } u_i \in \mathcal{A}^c, \\ \frac{1+\operatorname{sgn}(L_{0,i})}{2} & \text{if } u_i \in \mathcal{A}, i \in \mathcal{E}_\omega, \\ \frac{1-\operatorname{sgn}(L_{0,i})}{2} & \text{otherwise.} \end{cases} \quad (4)$$

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.

Let

$$p_i^*(\mathcal{E}_{\omega-1}) = \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_i = u_i | \boldsymbol{y}, \hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]_0^{i-1} = \boldsymbol{u}_0^{i-1}), \quad (5)$$

where $\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]_0^{i-1} = \{\hat{u}[\mathcal{E}_{\omega-1}]_0, \hat{u}[\mathcal{E}_{\omega-1}]_1, \ldots, \hat{u}[\mathcal{E}_{\omega-1}]_{i-1}\}$ and $\boldsymbol{u}_0^{i-1} = \{u_0, u_1, \ldots, u_{i-1}\}$. The probability that SC decoding is successful after flipping all the bit-flipping positions in $\mathcal{E}_\omega$ is then defined as [8]

$$P^*(\mathcal{E}_\omega) = \prod_{\substack{\forall i \in \mathcal{A} \setminus \mathcal{E}_\omega \\ i < i_\omega}} p_i^*(\mathcal{E}_{\omega-1}) \times \prod_{\forall i \in \mathcal{E}_\omega} (1 - p_i^*(\mathcal{E}_{\omega-1})). \quad (6)$$

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

$$i_\omega^* = \arg\max_{\substack{\forall i_\omega, i_{\omega-1} < i_\omega \leq N-1, u_{i_\omega} \in \mathcal{A} \\ \mathcal{E}_\omega = \mathcal{E}_{\omega-1} \cup i_\omega}} P^*(\mathcal{E}_\omega). \quad (7)$$

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 u are unknown to the decoder [8]. As a result, DSCF uses a known probability $p_i(\mathcal{E}_{\omega-1})$ to estimate $p_i^*(\mathcal{E}_{\omega-1})$, which is defined as

$$
\begin{aligned}
p_i(\mathcal{E}_{\omega-1}) &= \max \big(\Pr(\hat{u}[\mathcal{E}_{\omega-1}]_i = 0 | y, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i-1}), \\
&\qquad \Pr(\hat{u}[\mathcal{E}_{\omega-1}]_i = 1 | y, \hat{u}[\mathcal{E}_{\omega-1}]_0^{i-1}) \big) \\
&= \frac{1}{1 + \exp\left(-|L[\mathcal{E}_{\omega-1}]_{0,i}|\right)},
\end{aligned} \quad (8)
$$

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 [8] that the estimation in (8) is not accurate. Therefore, [8] introduced a perturbation parameter α to have a better estimation of $p_i^*(\mathcal{E}_{\omega-1})$ as

$$p_i^*(\mathcal{E}_{\omega-1}) \approx \frac{1}{1 + \exp\left(-\alpha |L[\mathcal{E}_{\omega-1}]_{0,i}|\right)}. \quad (9)$$

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 [8].

In order to have numerically stable computations in the hardware implementation of the DSCF decoder, the bit-flipping metric in (6) can be written in the log-likelihood (LL) domain as

$$
\begin{aligned}
Q^*(\mathcal{E}_\omega) &= -\frac{1}{\alpha} \ln(P^*(\mathcal{E}_\omega)) \\
&= \sum_{\substack{\forall i \in \mathcal{A} \\ i \leq i_\omega}} \frac{1}{\alpha} \ln\left(1 + \exp\left(-\alpha |L[\mathcal{E}_{\omega-1}]_{0,i}|\right)\right) \\
&\qquad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|.
\end{aligned} \quad (10)
$$

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

$$i_\omega^* = \arg\min_{\substack{\forall i_\omega, i_{\omega-1} < i_\omega \leq N-1, u_{i_\omega} \in \mathcal{A} \\ \mathcal{E}_\omega = \mathcal{E}_{\omega-1} \cup i_\omega}} Q^*(\mathcal{E}_\omega). \quad (11)$$

III. NEURAL DYNAMIC SUCCESSIVE CANCELLATION FLIP DECODING

Efficient hardware implementation of DSCF decoding is based on efficient implementation of the bit-flipping metric in (10). However, (10) involves logarithmic and exponential functions which are not hardware friendly. A common approach to
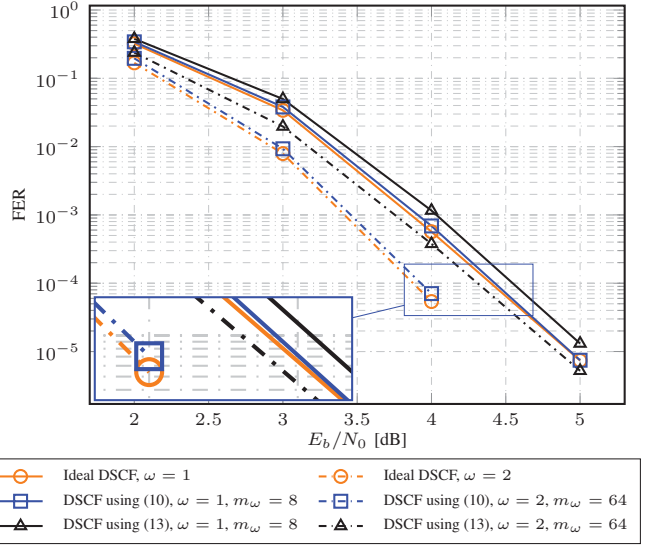

Fig. 2: Effect of the simplification in (13) on the FER of DSCF decoding for $\mathcal{P}(256, 128)$, concatenated with a 24-bit CRC. The ideal DSCF decoder is also plotted as a reference.

approximate the logarithmic and exponential function in (10) is to use the rectifier linear unit (ReLU) as [9]

$$\ln(1 + \exp(x)) \approx \text{ReLU}(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

However, since $\alpha > 0$, $-\alpha |L[\mathcal{E}_{\omega-1}]_{0,i}| < 0$. Therefore, (10) can be simplified as

$$
\begin{aligned}
Q^*(\mathcal{E}_\omega) &\approx \sum_{\substack{\forall i \in \mathcal{A} \\ i \leq i_\omega}} \frac{1}{\alpha} \text{ReLU}\left(-\alpha |L[\mathcal{E}_{\omega-1}]_{0,i}|\right) \\
&\qquad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}| \\
&= \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|,
\end{aligned} \quad (13)
$$

which is independent of the perturbation parameter α. Fig. 2 shows the effect of the simplification in (13) on the error-correction performance of DSCF decoding in terms of frame error rate (FER) for $\mathcal{P}(256, 128)$ concatenated with a 24-bit CRC which is used in 5G standard [2]. In this figure, $\alpha = 0.3367$ and is calculated as described in [8]. The FER of the ideal DSCF decoder where the erroneous bits up to the ω-th error order can always be accurately corrected is also plotted for comparison. It can be seen that using (13) incurs 0.1 dB and 0.4 dB of FER performance loss for DSCF decoding in comparison with using (10), when $\omega = 1$ and $\omega = 2$ respectively, at a target FER of 10^{-4}.

The main reason that results in such an error-correction performance degradation is that the perturbation parameter α is a multiplicative positive parameter which renders the ReLU function to be zero independent of the LLR value. To address this issue, we propose to use a perturbation parameter $\beta \in \mathbb{R}^+$ which 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

$$p_i^*(\mathcal{E}_{\omega-1}) \approx \frac{1}{1 + \exp\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right)}. \quad (14)$$

In addition, we propose to use a bit-flipping metric in the LL domain $Q^*(\mathcal{E}_\omega)$ which is tailored to the proposed $p_i^*(\mathcal{E}_{\omega-1})$ in (14) as

$$
\begin{aligned}
Q^*(\mathcal{E}_\omega) &= -\ln(P^*(\mathcal{E}_\omega)) + \omega\beta \\
&= \sum_{\substack{\forall i \in \mathcal{A} \\ i \leq i_\omega}} \ln\left(1 + \exp\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right)\right) \\
&\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|,
\end{aligned} \quad (15)
$$

where we used the fact that $\omega\beta$ is a constant and it will not affect the selection of i_ω^* in (11).

Let us now use the ReLU function in (12) to simplify the proposed bit-flipping metric in (15) as

$$
\begin{aligned}
Q^*(\mathcal{E}_\omega) &\approx \sum_{\substack{\forall i \in \mathcal{A} \\ i \leq i_\omega}} \mathrm{ReLU}\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right) \\
&\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}| \\
&= \sum_{\substack{\forall i \in \mathcal{A} \\ i \leq i_\omega \\ |L[\mathcal{E}_{\omega-1}]_{0,i}| < \beta}} \beta - |L[\mathcal{E}_{\omega-1}]_{0,i}| \\
&\quad + \sum_{\forall i \in \mathcal{E}_\omega} |L[\mathcal{E}_{\omega-1}]_{0,i}|,
\end{aligned} \quad (16)
$$

where we used the fact that if $\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}| > 0$, then $\mathrm{ReLU}\left(\beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|\right) = \beta - |L[\mathcal{E}_{\omega-1}]_{0,i}|$. It can be seen that the resulting hardware-friendly bit-flipping metric is dependent on the value of β.

We now show how to select the value of β. In conventional DSCF decoding of [8], finding the optimal value of α in (10) was treated as a search problem and was solved by a Monte-Carlo simulation. In this paper, we propose to treat the optimization of β in (15) or (16) as a learning problem and therefore, call the proposed algorithm NDSCF decoding. In order to do this, we use the inherent symmetric properties of the conventional DSCF decoding algorithm, which greatly simplify the training process as observed in [10]–[12].

Note that since $p_i(\mathcal{E}_{\omega-1})$ is independent from $\boldsymbol{u}$, the perturbed formulations of $p_i^*(\mathcal{E}_{\omega-1})$ in (9) and (14) are also independent from $\boldsymbol{u}$. Consequently, $P^*(\mathcal{E}_\omega)$ and $Q^*(\mathcal{E}_\omega)$ are also independent from $\boldsymbol{u}$. Therefore, all-zero codewords can be used for the training of β in (15) or (16). In addition, unlike the well-known classification problem in supervised learning [13], obtaining the output labels for the correct bit-flipping positions i_ω^* is not mandatory in the proposed framework. This is because if the estimation of bit i_ω^* is flipped to have a correct value for the next SC decoding attempt, the message word given by the next SC decoding attempt will have a correct value at the position i_ω^*, which is known to be 0
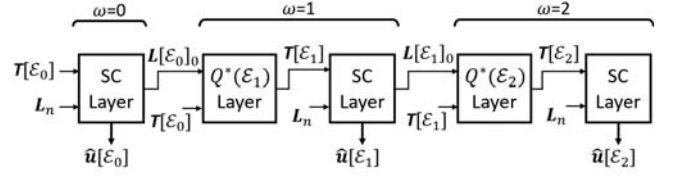

Fig. 3: NDSCF decoder training framework with $\omega \in \{0,1,2\}$.

since all-zero codewords are used. Instead of calculating an estimation error between i_ω^* and the true flipping position, we propose to use the output of SC decoding to calculate the objective loss function. Therefore, the heavy tasks of collecting the true flipping labels and representing them as one-hot encoded vectors are eliminated. As no labeling task is required for the error bits and only all-zero codewords are needed during the training, the proposed framework can be used in applications which require channel adaptation, where the decoder can be trained online to adapt with dynamic changes of the communication channel.

Fig. 3 illustrates the training framework of the proposed NDSCF decoder with $\omega \in \{0,1,2\}$. The framework contains two types of network layers, namely, SC and $Q^*(\mathcal{E}_\omega)$ layers. The SC layer performs standard SC decoding and the bit-flipping operations for all the bit indices in $\mathcal{E}_\omega$. On the other hand, the $Q^*(\mathcal{E}_\omega)$ layer is in charge of estimating the next bit-flipping position given that the estimated message word of the SC layer at error order $\omega - 1$ does not satisfy the CRC. However, as we use a batch of channel LLR values $\boldsymbol{L}_n$ for training, the decoding of different $\boldsymbol{L}_n$ instances can be terminated at different error orders. This problem complicates the implementation of the training framework with recent deep learning libraries [14]. To address this problem, we treat the decoding of all the $\boldsymbol{L}_n$ instances as the worst case scenario, where the correct message word can only be obtained after fixing the erroneous bit of the maximum considered error order. If a message word satisfies the CRC before reaching the last SC layer, the remaining $Q^*(\mathcal{E}_\omega)$ layers will maintain all the bit-flipping decisions of the successful CRC verification. Therefore, we define a bit-flipping vector $\boldsymbol{T}[\mathcal{E}_\omega] = \{T[\mathcal{E}_\omega]_0, \ldots, T[\mathcal{E}_\omega]_{N-1}\}$, which stores the bit-flipping decisions of all the information bits for the ω-th SC layer, with $T[\mathcal{E}_\omega]_i \in \{-1, +1\}$ such that a value of -1 indicates a bit flip and a value of $+1$ indicates no bit flip.

The training framework starts with the initial SC decoding at $\omega = 0$, which estimates the message word $\hat{\boldsymbol{u}}[\mathcal{E}_0]$ and the information bit LLR values $\boldsymbol{L}[\mathcal{E}_0]_0$. Note that $|\mathcal{E}_0| = \emptyset$ and $\boldsymbol{T}[\mathcal{E}_0] = \mathbf{1}$ where $\mathbf{1}$ is the all-ones vector of length N. If $\hat{\boldsymbol{u}}[\mathcal{E}_0]$ satisfies the CRC, the $Q^*(\mathcal{E}_1)$ and the $Q^*(\mathcal{E}_2)$ layers will not alter the bit-flipping vector and therefore set $\boldsymbol{T}[\mathcal{E}_1] = \mathbf{1}$ and $\boldsymbol{T}[\mathcal{E}_2] = \mathbf{1}$, respectively. As a result, $\hat{\boldsymbol{u}}[\mathcal{E}_1]$ and $\hat{\boldsymbol{u}}[\mathcal{E}_2]$ will have the same decoding values as $\hat{\boldsymbol{u}}[\mathcal{E}_0]$. On the other hand, if $\hat{\boldsymbol{u}}[\mathcal{E}_0]$ does not satisfy the CRC, given the input $\boldsymbol{L}[\mathcal{E}_0]_0$, the $Q^*(\mathcal{E}_1)$ layer computes either (15) or (16) for all the bit indices i_1 of all the candidates, where $0 \leq i_1 \leq N - 1$ and $u_{i_1} \in \mathcal{A}$. The $Q^*(\mathcal{E}_1)$ layer then selects i_1^* based on (11) and forms the

bit-flipping vector $\boldsymbol{T}[\mathcal{E}_1]$ as

$$T[\mathcal{E}_1]_i = \begin{cases} +1 & \text{if } 0 \le i \le N-1, \, i \ne i_1^*, \\ -1 & \text{if } i = i_1^*, \end{cases} \quad (17)$$

The bit-flipping vector $\boldsymbol{T}[\mathcal{E}_1]$ is then fed into the SC layer at error order $\omega = 1$, which performs a standard SC decoding given the channel LLR values $\boldsymbol{L}_n$ and estimates $\boldsymbol{u}[\mathcal{E}_1]$ based on $\boldsymbol{T}[\mathcal{E}_1]$ as

$$\hat{u}[\mathcal{E}_1]_i = \begin{cases} \frac{1-\text{sgn}(L[\mathcal{E}_1]_{0,i})T[\mathcal{E}_1]_i}{2} & \text{if } u[\mathcal{E}_1]_i \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

The decoding then continues in the same manner for all the layers at error order $\omega = 2$. It is worth mentioning that the SC layer is implemented by unfolding the decoding process specified in (1), (2), and (18) at all the required stages and bit indices.

By induction, we can write general expressions for any error order ω for $\boldsymbol{T}[\mathcal{E}_\omega]$ as

$$T[\mathcal{E}_\omega]_i = \begin{cases} T[\mathcal{E}_{\omega-1}]_i & \text{if } 0 \le i \le i_{\omega-1}, \\ +1 & \text{if } \text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]) = 0 \text{ and} \\ & i_{\omega-1} < i \le N-1, \\ +1 & \text{if } \text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]) \ne 0 \text{ and} \\ & i_{\omega-1} < i \le N-1, i \ne i_\omega^*, \\ -1 & \text{if } \text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]) \ne 0 \text{ and} \\ & i = i_\omega^*, \end{cases} \quad (19)$$

and for $\hat{\boldsymbol{u}}[\mathcal{E}_\omega]$ as

$$\hat{u}[\mathcal{E}_\omega]_i = \frac{1 - \text{sgn}(L[\mathcal{E}_\omega]_{0,i})T[\mathcal{E}_\omega]_i}{2}, \quad (20)$$

where $\text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]) = 0$ indicates a successful CRC verification, and $\text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]) \ne 0$ indicates a failed CRC verification for $\hat{\boldsymbol{u}}[\mathcal{E}_{\omega-1}]$.

As all the network layers enable back-propagation, β is trained with stochastic gradient descent technique [13]. In this paper, the following objective loss function is used:

$$\lambda = \sum_{\omega=1}^{2} l_\omega, \quad (21)$$

where

$$l_\omega = \begin{cases} (\hat{u}[\mathcal{E}_\omega]_{t_\omega} - u_{t_\omega})^2 & \text{if } \text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_\omega]) \ne 0, \, i_\omega^* \ge t_\omega, \\ 0 & \text{otherwise,} \end{cases}$$
$$\approx \begin{cases} \frac{1}{(1+\exp(L[\mathcal{E}_\omega]_{0,t_\omega}))^2} & \text{if } \text{CRC}(\hat{\boldsymbol{u}}[\mathcal{E}_\omega]) \ne 0, \, i_\omega^* \ge t_\omega, \\ 0 & \text{otherwise,} \end{cases} \quad (22)$$

and t_ω is the first erroneous bit position of $\hat{\boldsymbol{u}}[\mathcal{E}_\omega]$ with $u_{t_\omega} = 0$ as all-zero codewords are used. It is worth mentioning that the loss function is designed to penalize the wrong estimation of i_ω^* from the $Q^*(\mathcal{E}_\omega)$ layers only, not to penalize the wrong estimation of the message word $\hat{\boldsymbol{u}}[\mathcal{E}_\omega]$ obtained from the SC layers.
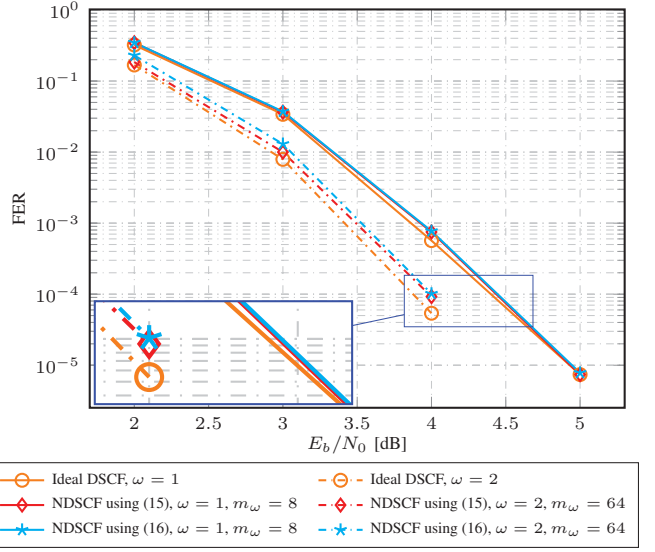

Fig. 4: FER of the proposed NDSCF decoding for $\mathcal{P}(256, 128)$, concatenated with a 24-bit CRC. The ideal DSCF decoder is also plotted as a reference.

TABLE I: Optimized parameter β of the proposed NDSCF decoders.

Metric	(15)		(16)	
ω	1	2	1	2
β	2.206	1.225	2.801	2.196

IV. EXPERIMENTAL RESULTS

In this section, the performance of the proposed NDSCF decoder is evaluated in terms of FER and the average number of decoding attempts. Throughout this section, $\mathcal{P}(256, 128)$ which is concatenated with a 24-bit CRC is considered for the evaluation, similar to the case in Fig. 2. The NDSCF decoders are trained with 2.5×10^5 samples of $\boldsymbol{L}_n$ at each E_b/N_0 value, where $E_b/N_0 \in \{2, 3, 4, 5\}$ dB. The parameter β at each error order ω is initialized with a uniform distribution within the interval of $(0, 10)$. The number of training epochs, mini-batch size, and learning rate are set to 50, 256, and 0.001, respectively. During the evaluation phase, each decoder is tested with at least 10^5 randomly generated codewords at each E_b/N_0 value, until at least 50 frames in error are captured.

Fig. 4 illustrates the FER of the proposed NDSCF decoder for $\omega \in \{1, 2\}$ and $m_\omega \in \{8, 64\}$. The FER of the ideal DSCF decoder where the erroneous bits up to the ω-th error order can always be accurately corrected is also plotted for comparison. Table I specifies the optimized parameter β for each bit-flipping metric in the proposed NDSCF decoder.

It can be observed from Fig. 4 that at a target FER of 10^{-4} and with $\omega = 1$, the proposed NDSCF decoders have almost the same error-correction performance compared to that of the corresponding ideal DSCF decoder. With $\omega = 2$ and at the target FER of 10^{-4}, the proposed NDSCF decoders
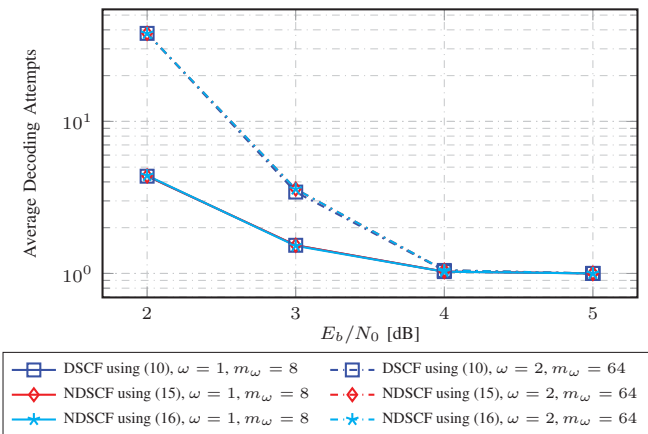
Fig. 5: Average number of decoding attempts of the proposed NDSCF decoders in comparison with that of the DSCF decoder of [8] for $\mathcal{P}(256, 128)$.

suffer from around 0.1 dB of error-correction performance loss in comparison with the corresponding ideal DSCF decoder. Nevertheless, it can be seen that the introduction of the hardware-friendly metric in (16) incurs almost no error-correction performance loss in comparison with using the metric in (15).

Fig. 5 depicts the average number of decoding attempts for the DSCF in [8] and the proposed NDSCF decoders. It can be seen that the average number of decoding attempts of the proposed NDSCF 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 Fig. 5 approaches 1 at high E_b/N_0 values. This indicates that at high E_b/N_0 values, the complexity of the decoders approaches the complexity of a single SC decoder.

V. CONCLUSION

In this paper, we proposed a neural dynamic successive cancellation flip (NDSCF) decoding algorithm of 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 decoding complexity approaches that of the successive cancellation (SC) decoding at high signal-to-noise ratio (SNR) regimes; (ii) only additions and multiplexers 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; and (iv) it enables channel adaptation and simplifies over-the-air training as no labeling task is required and only all-zero codewords are needed during the training process. With the aforementioned advantages, NDSCF decoding is a potential candidate for practical applications in the 5G standard.

VI. ACKNOWLEDGMENT

The authors would like to thank Arash Ardakani and Adam Cavatassi of McGill University for their helpful and construc-

REFERENCES

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

[2] 3GPP, "Multiplexing and channel coding (Release 10) 3GPP TS 21.101 v10.4.0." Oct. 2018. [Online]. Available: http://www.3gpp.org/ftp/Specs/2018-09/Rel-10/21_series/21101-a40.zip

[3] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, March 2015.

[4] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *48th Asilomar Conf. on Sig., Sys. and Comp.*, Nov 2014, pp. 2116–2120.

[5] F. Ercan, C. Condo, S. A. Hashemi, and W. J. Gross, "Partitioned successive-cancellation flip decoding of polar codes," *arXiv e-prints*, p. arXiv:1711.11093v4, Nov 2017. [Online]. Available: https://arxiv.org/abs/1711.11093

[6] C. Condo, F. Ercan, and W. J. Gross, "Improved successive cancellation flip decoding of polar codes based on error distribution," in *IEEE Wireless Commun. and Net. Conf. Workshops*, April 2018, pp. 19–24.

[7] F. Ercan, C. Condo, and W. J. Gross, "Improved bit-flipping algorithm for successive cancellation decoding of polar codes," *IEEE Trans. on Commun.*, vol. 67, no. 1, pp. 61–72, Jan 2019.

[8] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2333–2345, June 2018.

[9] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.

[10] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. of Sel. Topics in Signal Process.*, vol. 12, no. 1, pp. 119–131, February 2018.

[11] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *IEEE Int Symp. on Inf. Theory*, August 2017, pp. 1361–1365.

[12] N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier, and W. J. Gross, "Neural belief propagation decoding of CRC-polar concatenated codes," in *IEEE Int. Conf. on Commun.*, May 2019, pp. 1–6.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, May 2015.

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Conf. on Operating Systems Design and Impl.*, ser. OSDI'16. USENIX Association, 2016, pp. 265–283.