DNC-Aided SCL-Flip Decoding of Polar Codes

Yaoyu Tao and Zhengya Zhang

Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109 USA
Emails: {taoyaoyu, zhengya}@umich.edu

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685277

***Abstract*—Successive-cancellation list (SCL) decoding of polar codes has been adopted for 5G wireless communications. However, the performance of moderate code length is not satisfactory. Heuristic or deep-learning-aided (DL-aided) flip algorithms have been developed to improve the performance by locating error bit positions after SCL decoding. In this work, we propose a new flip algorithm with the help of differentiable neural computer (DNC). New state and action encoding are developed to improve DNC training and inference efficiency. The proposed two-phase method is done by a flip DNC (F-DNC) to rank the most likely flip positions for multi-bit flipping, and if decoding still fails, a flip-validate DNC (FV-DNC) is applied to re-select error bit positions in successive flip decoding trials. Supervised training methods are designed for the two DNCs. Simulation results show that the proposed DNC-aided SCL-Flip (DNC-SCLF) decoding demonstrates up to 0.34 dB coding gain or 54.2% reduction in the average number of decoding attempts over prior work.**

***Index Terms*—Polar code, deep learning, successive cancellation list decoder, flip algorithms, differentiable neural computer**

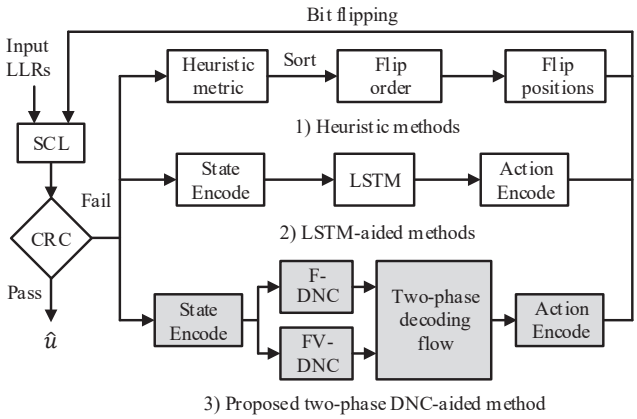

Fig. 1. Overview of 1) heuristic bit flipping, 2) LSTM-aided bit flipping and 3) proposed DNC-aided two-phase bit flipping.

I. Introduction

Capacity-achieving polar codes [1] have been adopted in the 5th generation (5G) wireless standard. They can be decoded sequentially on a trellis using a successive cancellation list (SCL) [2] decoder. Upon receiving log-likelihood ratios (LLRs), SCL calculates path metrics (PMs) following a bit-by-bit order. A list of L most likely paths are kept during decoding and bits are decoded based on the most likely path that also passes a cyclic redundancy check (CRC) in the end. However, the decoding performance is not satisfactory for moderate code length N. Once a wrong bit decision is made, the entire sequential decoding fails.

To solve this problem, flip algorithms are used when CRC fails. Error bit positions are searched and flipped in subsequent decoding attempts. Clearly, the key to a successful flip decoding is to accurately locate error bit positions. As shown in Fig. 1, heuristic methods [3]–[16] use explicit metrics to estimate the likelihood of each bit being an error bit. The likelihoods are sorted to obtain the flip position set. However, the optimal flipping strategy is still an open problem.

Recent work on flip algorithms has leveraged deep learning (DL). DL-aided methods require state encoding to pre-process the inputs to the neural network (NN) and action encoding to generate flip position set from the NN outputs, as shown in Fig. 1. [7], [17]–[19] propose to use long short-term memory (LSTMs) to help locate flip positions for short polar codes of length 64 or 128. However, LSTMs lack the scalability to handle long-distance dependencies embedded in the sequential SCL decoding when code length increases, presenting a limitation for practical adoptions.

The recently developed differentiable neural computer (DNC) [20] addresses the scaling problem of LSTM. DNC can be considered an LSTM augmented with an external memory through attention-based soft read/write mechanisms. In this paper, we propose to use DNC for bit flipping of practical-length polar codes to enhance the accuracy of locating error bit positions. The main contributions are summarized as follows:

1) A new two-phase decoding assisted by two DNCs, flip DNC (F-DNC) and flip-validate DNC (FV-DNC), as shown in Fig. 1. F-DNC ranks the most likely flip positions for multi-bit flipping. If decoding still fails, FV-DNC is used to re-select flip positions in successive flip decoding trials.
2) A new action encoding with soft multi-hot scheme and a new state encoding considering both PMs and received LLRs for a higher DNC training and inference efficiency. Training methods are designed for the two DNCs, where the training data set is generated by supervised flip decoding attempts.
3) The effectiveness of DNC-aided SCL-Flip (DNC-SCLF) decoder is demonstrated to outperform the state-of-the-art techniques by up to 0.34 dB in error correction performance or 54.2% reduction in the average number of decoding attempts.

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

II. Background

A. SCL Decoding of Polar Codes

An $(N,\ K)$ polar code has a code length N and code rate K/N. Let $u_0^{N-1} = (u_0, u_1, ..., u_{N-1})$ denote the vector of input bits to the encoder. The K most reliable bits in u_0^{N-1}, called free bits, are used to carry information; while the remaining $N-K$ bits, called frozen bits, are set to pre-determined values.

Successive cancellation (SC) [1] is the basic decoding scheme of polar codes. Assume r_0^{N-1} is the received LLRs. The decoding follows a bit-by-bit sequential order and it calculates bit LLR $L^{\hat{u}_i}$ for i-th bit on the SC trellis, where $i = \{0, ..., N-1\}$ and $\hat{u}_i = \pm 1$. The decoding of a bit depends on the previously decoded bits. In SC decoding, the most likely path to each bit level is kept. SCL decoding [2] improves the error-correction performance by keeping a list of L mostly likely paths based on the PM $\mathcal{P}(\ell)_i$ to each bit level, where ℓ and i denote the path index and the bit index, respectively. For each path ℓ and each bit i, the PMs are defined in (1):

$$\mathcal{P}(\ell)_i \triangleq \sum_{j=0}^{i} \ln(1 + e^{-(1-2\hat{u}_j(\ell))L^{\hat{u}_j}(\ell)}), \tag{1}$$

where $\hat{u}_j(\ell)$ and $L^{\hat{u}_j}(\ell)$ denote the j-th bit at ℓ-th path and the bit LLR for $\hat{u}_j$ given the received LLRs r_0^{N-1} and decoding trajectories $\hat{u}_0^{j-1}(\ell)$, respectively. SC can be viewed as a special case when the list size $L = 1$. Concatenating polar code with a CRC [21], [22] can aid the final path selection.

B. State-of-the-art Flip Algorithms

Flip algorithms are proposed to identify error bit positions upon failed CRC. The flip positions can be determined by either a heuristic metric or an NN. Heuristic methods like [3]–[6], [9] use received LLRs or their absolute values as the metric to derive flip positions. In particular, [6] introduces a critical set to reduce the search space of flip positions for a lower complexity. [9] subdivides a codeword into partitions, on which SC-Flip (SCF) is run for a shorter latency. However, these methods can only flip one bit at a time. [10], [11], [13], [14] propose a dynamic SC-Flip (DSCF) that allows flipping of multiple bits at a time to improve the latency of SCF. Multi-bit flipping requires locating multiple error bit positions concurrently. DSCF introduces a new metric considering not only received LLRs but also the trajectories in the sequential SCL decoding. [13], [14] introduce variations of DSCF to improve the accuracy of locating error bit positions. [8], [16] extend the bit-flipping from SC to SCL for a SCL-Flip decoding (SCLF).

The recently developed DL-aided SCF/SCLF [7], [17]–[19] utilize a trained LSTM to locate error bit positions. They have shown slightly better performance than heuristic methods for short polar codes of length 64 or 128. However, the accuracy of locating error bit positions is limited by the scalability of LSTMs when the code length increases. Furthermore, the LSTM methods use simple state and action encoding that do not support multi-bit flipping efficiently, resulting in more decoding attempts compared to heuristic methods.

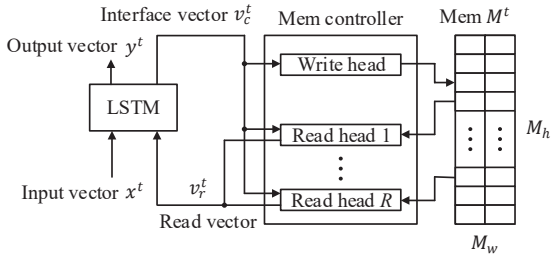

Fig. 2. Top-level architecture of DNC.

C. Differentiable Neural Computer (DNC)

DNC addresses LSTM's scalability problem with the help of an external memory. Since its invention, DNC has found many applications like question answering [23], [24]. DNC can be considered an LSTM augmented with an external memory through soft read and write heads, as shown in Fig. 2. In this work, we use DNCs to enhance the accuracy of locating error bit positions.

The top level architecture of DNC is demonstrated in Fig. 2. DNC periodically receives x^t as the input vector and produces y^t as the output vector at time t. The output vector y^t is usually converted to a probability distribution using softmax. At time t, the DNC 1) reads an input x^t; 2) writes the new information into the external memory using the interface vector v_c^t through a memory controller; 3) reads the updated memory M^t; and 4) produces an output y^t. Assume the external memory is a matrix of M_h slots and each slot is a length-M_w vector. To interface with this external memory, DNC computes read and write keys to locate slots. The memory slot is found using the similarity between a key and the slot content. This mechanism is known as content-based addressing. In addition, DNC uses dynamic memory allocation and temporal memory linkage mechanisms to compute write and read weights. We omit the mathematical descriptions of DNC here and readers can refer to [20] for more details.

III. DNC-Aided Flip Decoding

Bit flipping can be modeled as a game and DNC is the player to locate flip positions towards successful decoding. Upon CRC failure, the DNC player needs to take an action based on the current state, either reverting falsely flipped positions or adding more positions to flip. The proposed DNC-aided method includes new state and action encoding and a DNC-aided two-phase decoding flow.

A. State and Action Encoding

One key to an efficient DNC inference is a suitable input (state) and output (action) vector for training and inference. We discuss the encoding of existing LSTM-based approaches [7], [17]–[19] and present a new encoding scheme.

1) State Encoding: A straightforward way to encode states is to use the received LLR sequence r_0^{N-1}. [7], [17] use the amplitudes of received LLRs as the LSTM input. [19] uses the amplitudes of received LLRs, combined with the syndromes generated by CRC for state encoding. However, the PM information in sequential decoding is not considered in these methods, resulting in a potential loss. [18] proposes a state encoding by taking the ratio of the PM of discarded paths to the PM of survival paths. However, this representation requires extra computation and does not include received LLR information.

In this work, we introduce a new state encoding scheme using the gradients of L survival paths concatenated with received LLRs. The PM gradients $\triangledown\mathcal{P}(\ell)_i$ for i-th bit is given by (2):

$$\triangledown\mathcal{P}(\ell)_i = \ln(1 + e^{-(1-2\hat{u}_i(\ell))L^{\hat{u}_i}(\ell)}). \tag{2}$$

Note that $\triangledown\mathcal{P}(\ell)_i$ can be directly taken from the PM calculation in standard SCL decoding. The state encoding S is therefore a vector shown in (3) and is used as DNC input in this work.

$$S = \{\triangledown\mathcal{P}(\ell)_0^{N-1}, r_0^{N-1}\}. \tag{3}$$

2) Action Encoding: the one-hot scheme used in state-of-the-art LSTM-based flip algorithms is efficient in locating the first error bit, but not multiple bits at a time. As a result, more decoding attempts are needed. To improve the bit flipping efficiency, we propose a soft multi-hot (i.e., ω-hot) flip vector v_f to encode both the first error bit and the subsequent error bits, aiming to correctly flip multiple bits in one attempt. v_f is a length-N vector that has ω non-zero entries. An action is encoded by v_f. Each possible flip position in v_f is a non-zero soft value indicating the flip likelihood of the bit.

For training purposes, we introduce a scaled logarithmic series distribution (LSD) to assign flip likelihoods to the ω flip positions, where $p \in (0, 1)$ is a shape parameter of LSD. The intention is to create a distribution with descending probabilities from the first error bit position to the subsequent ones, and to provide enough likelihood differences between them. Suppose the k-th bit in polar code has an index $\mathcal{I}_{\mathcal{F}}(k)$ in the flip position set $\mathcal{F}$. Non-zero entries of v_f can be derived in (4):

$$\begin{aligned} v_f(k) &= \mathcal{K}\frac{-1}{\ln(1-p)}\frac{p^{\mathcal{I}_{\mathcal{F}}(k)}}{\mathcal{I}_{\mathcal{F}}(k)} \text{ for } k \in \mathcal{F} \\ \text{where scaling factor } \mathcal{K} &= 1/\int_{\mathcal{F}} v_f \end{aligned} \tag{4}$$

Reference v_f generation for training will be discussed in Section IV. The impacts of parameters ω and p on the accuracy of locating error bit positions are discussed in Section V-A.

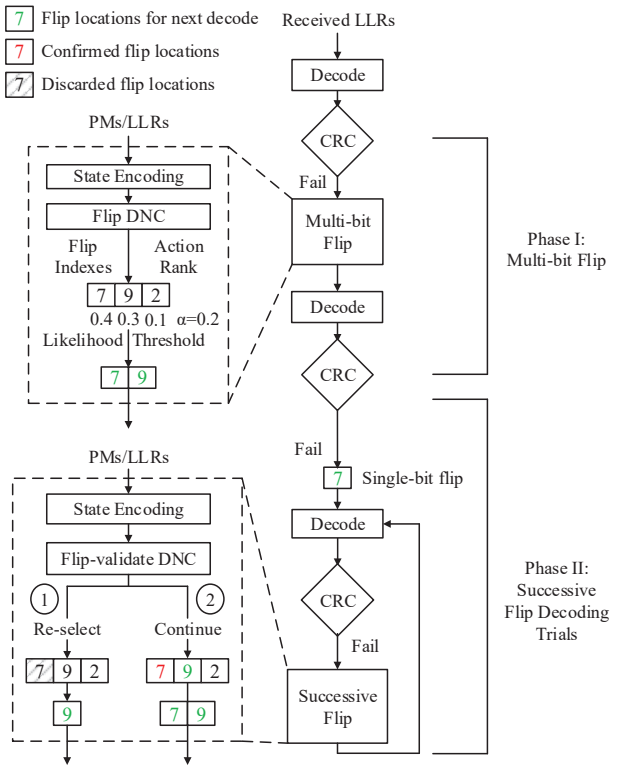

Fig. 3. DNC-aided two-phase flip decoding ($\omega = 3$ case).

B. DNC-Aided Two-Phase Decoding Flow

We design a two-phase flip decoding flow aiming to reduce the number of SCL attempts while achieving a good error correction performance. The two phases in this flow are: 1) multi-bit flipping and 2) successive flip decoding trials. In the first phase, the received symbols are decoded by a standard decoder. If it fails CRC, a flip DNC (F-DNC) takes as input the state encoding S to score the actions, i.e., estimate the probability of each bit being error bits and output a flip vector v_f. Fig. 3 shows an example of $\omega = 3$ where $\mathcal{F} = \{7, 9, 2\}$ is the flip position set with descending likelihoods $\{0.4, 0.3, 0.1\}$. To avoid wrong flips of subsequent positions with insignificant flip likelihoods, an α-thresholding is applied to keep only positions with $v_f(i) > \alpha, i = \{0, ..., N-1\}$. A subsequent decode attempt is then carried out with multi-bit flipping of bit positions $\{7, 9\}$ in the example.

Phase-I decoding can fail for several reasons: 1) the first error bit position is incorrect; or 2) the first error bit position is correct but some of the subsequent flip positions are incorrect. If CRC still fails after multi-bit flipping in Phase I, we enter Phase II to flip each possible error bit position one at a time and use a flip-validate DNC (FV-DNC) to confirm if this is a correct flip before moving to the next error bit position. The first attempt in Phase II flips the highest ranked error bit position in $\mathcal{F}$, i.e., bit 7 in the example shown in Fig. 3.

If FV-DNC invalidates the single-bit flip (e.g., bit 7 in this example), we discard bit 7 and re-select the next bit, bit 9 in $\mathcal{F}$, as the flip position. Alternatively, if FV-DNC confirms

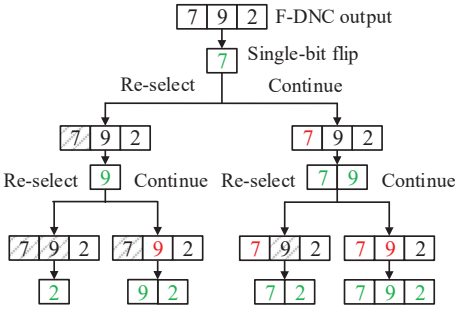

Fig. 4. Possible flip attempts in Phase II ($\omega = 3$).

the flip of bit 7, we continue by adding bit 9 into the flip queue $\mathcal{Q}_f$ and flip $\mathcal{Q}_f = \{7, 9\}$ in next attempt. The process runs successively until CRC passes or the end of $\mathcal{F}$ is reached. Fig. 4 shows all possible flip combinations given different FV-DNC output combinations in the $\omega = 3$ case. The number of decoding attempts of Phase II is bounded by ω. The two-phase DNC-SCLF is described in Algorithm 1.

Algorithm 1: DNC-Aided SCL-Flip Decoding

$\hat{u}_0^{N-1}, S \leftarrow$ SCL(r_0^{N-1})
if CRC$(\hat{u}_0^{N-1}) =$ pass **return** $\hat{u}_0^{N-1}$
Phase-I: Multi-bit Flipping
$\mathcal{F}, \omega, v_f \leftarrow$ F-DNC(S)
$\hat{u}_0^{N-1} \leftarrow$ SCL$(r_0^{N-1}, \mathcal{F}_{v_f \geq \alpha})$
if CRC$(\hat{u}_0^{N-1}) =$ pass **return** $\hat{u}_0^{N-1}$
Phase-II: Successive Flip Decoding Trials
$\mathcal{Q}_f = \{\mathcal{F}[0]\}$
for $i = 0, 1, ..., \omega - 1$ **do**
  $\hat{u}_0^{N-1}, S \leftarrow$ SCL$(r_0^{N-1}, \mathcal{Q}_f)$
  if CRC$(\hat{u}_0^{N-1}) =$ pass or $i = \omega - 1$ **return** $\hat{u}_0^{N-1}$
  $\mathcal{R} \leftarrow$ FV-DNC(S)
  if $\mathcal{R} =$ continue **then**
    $\mathcal{Q}_f = \{\mathcal{Q}_f, \mathcal{F}[i+1]\}$
  else
    $\mathcal{Q}_f[\text{end}] = \mathcal{F}[i+1]$
  end
end

IV. Training Methodology

In this section, we discuss training for the DNCs used in proposed DNC-SCLF. The training is conducted offline and does not increase the run-time decoding complexity. We adopt the cross-entropy function which has been widely used in classification tasks [25].

A. F-DNC Training

In the first training stage, we run extensive SCL decoder simulations and collect error frames upon CRC failure. The F-DNC training data set consists of state encoding S from (3) as DNC input and the corresponding v_f from (4) as the reference output. S can be straightforwardly derived based on received LLRs and PMs of the collected error frames. v_f is determined by parameter ω and p, whose values affect the training and inference efficiency. We label the error bit positions with respect to the transmitted sequence for each sample as candidate flip positions. Intuitively, small ω and p values increase the likelihood of locating the first error bit position, but decrease the likelihoods of locating subsequent error bit positions. Hence a trade-off exists between the accuracy of locating the first error bit position and the subsequent error bit positions. In this work, we carried out reference v_f generations with $\omega = \{2, 5, 10\}$ and $p = \{0.2, 0.8\}$. The experimental results using these parameters will be discussed in Section V.

TABLE I
F-DNC/FV-DNC Hyper-parameters Set

Parameter	Description
LSTM controller	1 layer of size 128
Size of access heads	1 write head, 4 read heads
Size of external memory	$M_h = 256, M_w = 128$
Size of training set	10^6 for F-DNC, 3×10^7 for FV-DNC
Size of validation set	5×10^4
Mini-batch size	100
Dropout probability	0.05
Optimizer	Adam
Environment	Tensorflow 1.14.0 on Nvidia GTX 1080Ti

B. FV-DNC Training

The error frames that cannot be decoded correctly in Phase I are handled in Phase II, where single bit positions are flipped and tested successively as shown in Fig. 4. The FV-DNC is a classifier taking either "re-select" or "continue" action given the received LLRs and PMs from the most recent attempt. The key to FV-DNC training is to create a curated data set that detects "trapping states" effectively. We carried out supervised flip decoding attempts based on the collected errors from Phase I. For each error, the first 5 error bit positions in the reference v_f are flipped bit after bit and their corresponding state encoding S is recorded. These samples result in a "continue" action. After flipping each of the first 5 error bit positions, we flip 5 random positions and record their state encoding S. These samples indicate trapping states and result in a "re-select" action. For each collected error frame, we produce 5 samples for "continue" action and 25 samples for "re-select" action.

V. Experiments and Analysis

To show the effectiveness of DNC in tackling long-distance dependencies in polar decoding trellis, we evaluate the performance of polar codes of length N = 256 and 1024 using SC and SCL ($L = 4$) decoding. The code rate is set to 1/2. A 16b CRC is employed. Error frames are collected at an SNR of 2 dB. In this paper we do not focus on hyper-parameter optimizations for DNC and simply demonstrate a set of configurations, shown in Table I, that work well for our experiments.

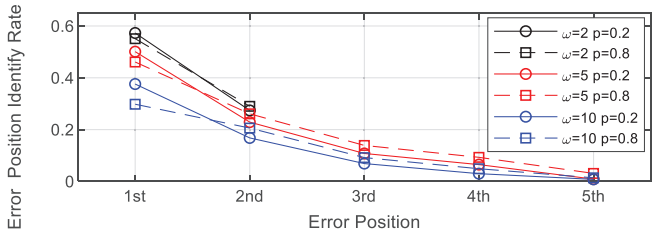

Fig. 5. Rate of locating error bit positions for $\omega = \{2, 5, 10\}$ and $p = \{0.2, 0.8\}$ in SC decoding of a (256, 128) polar code.

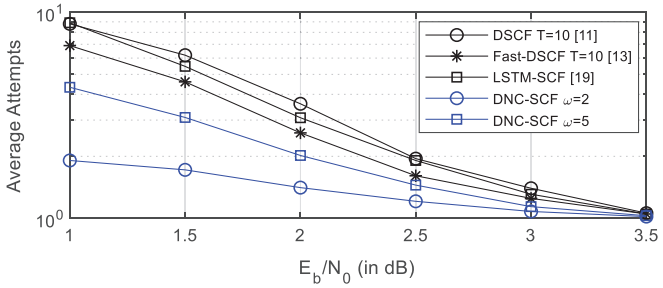

Fig. 6. Number of additional decoding attempts using DNC-SCF and state-of-the-art flipping algorithms for a (1024, 512) polar code.

A. Accuracy of Locating Error Bits

First, we study the impacts of parameters ω and p in action encoding. For a fair comparison, we pick the same code length of N = 256 and SC decoding used by the heuristic method [11] and the LSTM-based method [17]. Fig. 5 presents the accuracy of locating the first 5 error bit positions. For a given ω, a lower p enhances the probability of locating the first error bit, but reduces the probability of locating subsequent error bits. We achieve a 0.573 success rate of locating the first error bit with $\omega = 2$, outperforming the 0.425 and 0.51 success rate with the heuristic DSCF [11] and the LSTM-aided SCF [17], respectively. Comparing $\omega = 2$ and $\omega = 5$ with the same p value, a larger ω helps to locate more error bit positions, but the success rate of locating the first few bit positions is degraded.

We select $p = 0.8$ in our two-phase DNC-SCLF experiments to increase the success rates of locating subsequent error bit positions by sacrificing the success rate of locating the first error bit position. Even if F-DNC may not locate the first error bit position accurately in Phase I, FV-DNC can re-select it in Phase II. We use an $\alpha = 0.03$ for thresholding in our experiments.

B. Complexity and Latency

Metric calculation and sorting in heuristic methods can be implemented inside standard SC or SCL decoders. However, DL-aided algorithms introduce a higher complexity and require an inference accelerator to interact with the decoder. We used a GPU to obtain a speed of 1.7 ms/inference. For practical adoptions, a dedicated accelerator can be implemented for a faster inference.

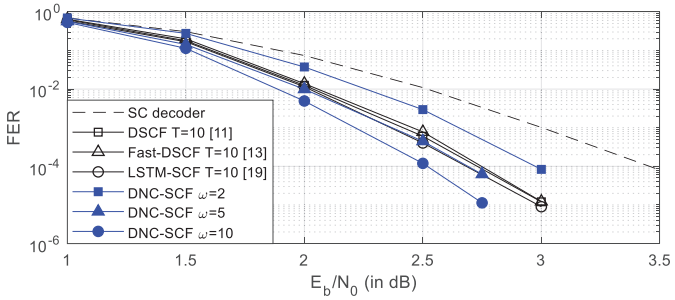

Fig. 7. FER performance comparison between DNC-SCF and state-of-the-art flipping algorithms for a (1024,512) polar code and a 16b CRC.

Since bit flipping is conditionally triggered when the standard decoder fails, the triggering rate is the frame error rate (FER). For this reason, the DL-aided algorithms are more suitable for the low-FER regime where the inference latency can be effectively amortized. In this work we do not consider the inference acceleration and buffering strategy, but instead focus on the average number of flip decoding attempts to determine the overall latency.

Assume β_1 is the rate of successful decoding with multi-bit flipping in Phase I, the average number of decoding attempts T_{avg} for a DNC-aided flip decoding can be calculated in (5):

$$T_{avg} = \beta_1 + \omega_{2,\text{avg}}(1 - \beta_1) \tag{5}$$

where $\omega_{2,\text{avg}}$ is the average number of attempts in Phase II and $\omega_{2,\text{avg}} \leq \omega$. Fig. 6 shows the T_{avg} for the proposed DNC-SCF along with the state-of-the-art techniques. At a 2 dB SNR, DNC-SCF with $\omega = 2$ improves the average decoding attempts by 45.7% and 54.2% compared to the state-of-the-art heuristic [13] and LSTM-aided methods [19], respectively.

C. Error-Correction Performance

We compare the coding gain of DNC-SCF at an FER of 10^{-4} with the state-of-the-art heuristic methods [11], [13] and LSTM-based methods [19] for a (1024, 512) polar code with a 16b CRC. DNC-SCF with $\omega = 2$ achieves a 0.5 dB coding gain over the SC decoder. Increasing to $\omega = 5$ provides another 0.31 dB coding gain. DNC-SCF with $\omega = 5$ outperforms DSCF [11] and Fast-DSCF [13] with $T = 10$ by 0.03 dB and 0.05 dB, respectively, while reducing the number of decoding attempts by 45.7%. Further increasing to $\omega = 10$ in DNC-SCF provides a 0.21 dB coding gain over DSCF with $T = 10$ while reducing the number of decoding attempts by 18.9%.

The LSTM-based approach in [17] does not report FER, but has shown up to 10% improvement in the accuracy of locating the first error bit position over DSCF with $T = 1$ at a 1 dB SNR for a (64, 32) polar code. Another LSTM-based SCF [19] with $T = 6$ provides a 0.2 dB improvement over DSCF with $T = 6$. The FER of [19] with $T = 10$ for a 1024b code is shown in Fig. 7, which is still worse than DNC-SCF with $\omega = 5$. LSTM's capability of locating error bit positions weakens as the code length increases.

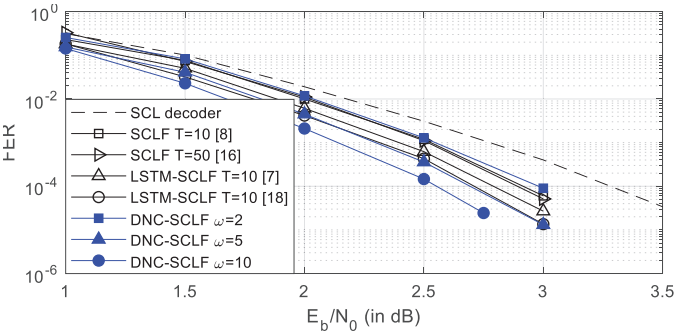

Fig. 8. FER performance comparison between DNC-SCLF ($L = 4$) and state-of-the-art flipping algorithms for a (256,128) polar code and a 16b CRC.

We further compare the FER of DNC-SCLF ($L = 4$) for a (256, 128) polar code with a 16b CRC to the state-of-the-art heuristic methods [8], [16] and LSTM-based approaches [7], [18] as shown in Fig. 8. At an FER of 10^{-4}, DNC-SCLF with $\omega = 2$ achieves a 0.27 dB coding gain over the SCL decoder. Increasing to $\omega = 5$ results in a 0.59 dB coding gain over the SCL decoder. DNC-SCLF with $\omega = 5$ achieves 0.21 dB and 0.01 dB better coding gain than the heuristic SCLF [16] and LSTM-SCLF [18] with $T = 10$, respectively. Further increasing to $\omega = 10$ in DNC-SCLF improves the coding gain to 0.34 dB and 0.16 dB over [16] and [18], respectively.

VI. Conclusions

We present DNC-aided SCF and SCLF decoding that employ two-phase decoding assisted by two DNCs, F-DNC to locate error bit positions for multi-bit flipping, and FV-DNC to re-select error bit positions for successive flip decoding trials, respectively. The multi-bit flipping reduces the number of flip decoding attempts while successive flip decoding trials lower the probability of entering a trapping state. Methods are proposed to efficiently train F-DNC and FV-DNC. Simulation results show that the proposed DNC-SCF and DNC-SCLF help to locate error bits more accurately, achieving better error-correction performance and reducing the number of flip decoding attempts than the the state-of-the-art flip algorithms.

References

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

[2] I. Tal and A. Vardy, "List decoding of polar codes," in *2011 IEEE International Symposium on Information Theory Proceedings*, July 2011, pp. 1–5.

[3] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, Nov 2014, pp. 2116–2120.

[4] C. Condo, F. Ercan, and W. Gross, "Improved successive cancellation flip decoding of polar codes based on error distribution," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr 2018, pp. 19–24.

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

[6] Z. Zhang, K. Qin, L. Zhang, H. Zhang, and G. T. Chen, "Progressive bit-flipping decoding of polar codes over layered critical sets," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[7] X. Liu, S. Wu, Y. Wang, N. Zhang, J. Jiao, and Q. Zhang, "Exploiting error-correction-crc for polar scl decoding: A deep learning based approach," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2019.

[8] F. Cheng, A. Liu, Y. Zhang, and J. Ren, "Bit-flip algorithm for successive cancellation list decoder of polar codes," *IEEE Access*, vol. 7, pp. 58 346–58 352, 2019.

[9] F. Ercan, C. Condo, S. A. Hashemi, and W. J. Gross, "Partitioned successive-cancellation flip decoding of polar codes," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[10] L. Chandesris, V. Savin, and D. Declercq, "An improved scflip decoder for polar codes," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[11] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-scflip decoding of polar codes," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2333–2345, June 2018.

[12] Y. Tao, S. G. Cho, and Z. Zhang, "A configurable successive-cancellation list polar decoder using split-tree architecture," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 2, pp. 612–623, 2021.

[13] F. Ercan, T. Tonnellier, N. Doan, and W. J. Gross, "Practical dynamic sc-flip polar decoders: Algorithm and implementation," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5441–5456, 2020.

[14] F. Ercan and W. J. Gross, "Fast thresholded sc-flip decoding of polar codes," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[15] C. Condo, V. Bioglio, and I. Land, "Sc-flip decoding of polar codes with high order error correction based on error dependency," in *2019 IEEE Information Theory Workshop (ITW)*, 2019, pp. 1–5.

[16] Y. H. Pan, C. H. Wang, and Y. L. Ueng, "Generalized scl-flip decoding of polar codes," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[17] X. Wang, H. Zhang, R. Li, L. Huang, S. Dai, Y. Yourui, and J. Wang, "Learning to flip successive cancellation decoding of polar codes with lstm networks," arXiv:1902.08394, Feb 2019.

[18] C.-H. Chen, C.-F. Teng, and A.-Y. Wu, "Low-complexity lstm-assisted bit-flipping algorithm for successive cancellation list polar decoder," in *45th IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2020.

[19] B. He, S. Wu, Y. Deng, H. Yin, J. Jiao, and Q. Zhang, "A machine learning based multi-flips successive cancellation decoding scheme of polar codes," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[20] A. Graves, G. Wayne, and M. Reynolds et al., "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, pp. 471–476, Oct 2016.

[21] K. Niu and K. Chen, "Crc-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, October 2012.

[22] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Communications Letters*, vol. 16, no. 12, pp. 2044–2047, December 2012.

[23] S. Sukhbaatar, a. szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2440–2448.

[24] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 20–22 Jun 2016, pp. 1378–1387.

[25] Y. Lecun, Y. Bengio, and G. Hinton et al., "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, Oct 2015.