Neural Successive Cancellation Decoding of Polar Codes

Nghia Doan, Seyyed Ali Hashemi, Warren J. Gross
Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada
Email: nghia.doan@mail.mcgill.ca, seyyed.hashemi@mail.mcgill.ca, warren.gross@mcgill.ca

***Abstract*—Neural network (NN) based decoders have appeared as potential candidates to replace successive cancellation (SC) based and belief propagation (BP) decoders for polar codes, due to their one-shot-decoding property. Partitioned NN (PNN) decoder has provided a solution to make use of multiple NN decoders which are connected with BP decoding, with the presence of insufficient training data for practical-length polar codes. However, PNN decoder requires BP iterations that detrimentally affect the decoding latency as compared to non-iterative approaches. In this paper, we propose a neural SC (NSC) decoder to overcome the issue associated with PNN. Unlike PNN, the NSC decoder is constructed by multiple NN decoders connected with SC decoding. Compared to a PNN decoder for a polar code of length 128 and rate 0.5, the proposed NSC decoder achieves the same decoding performance, while reducing the decoding latency by 42.5%.**

***Index Terms*—polar codes, successive-cancellation decoding, list decoding, belief propagation decoder, deep-learning-based decoders.**

I. Introduction

Polar codes, proposed by Arıkan in [1], are a recent breakthrough in coding theory owing to the fact that they represent a class of error-correcting codes that is mathematically proven to achieve channel capacity with low complexity encoding and decoding algorithms. Successive cancellation (SC) and belief propagation (BP) decoding algorithms are two methods used to decode polar codes. SC decoding has low computational complexity, but its serial nature limits the throughput of decoding. In addition, polar codes only achieve channel capacity under SC decoding when the code length is very high. For short to moderate code lengths, SC decoding fails to provide a reasonable error correction performance. SC list (SCL) decoding [2] improves the error correction performance of SC decoding at the cost of higher computational complexity and lower throughput. Several attempts have been made to reduce the computational complexity and increase the throughput of polar code SCL decoders [3]–[5].

Although the iterative message-passing characteristic of BP decoding enables parallel decoding, this algorithm requires high computational complexity and is not able to achieve desirable error correction performance within a small number of iterations [6]. Recently, deep learning (DL) algorithms and specifically neural networks (NNs), have been considered as a potential solution to either replace or help the state-of-the-art decoders for polar codes [7]–[14] because of their powerful prediction models and one-shot-decoding property [14]. The latter characteristic is especially useful since polar codes are selected as a channel coding scheme for the next generation of wireless communications standard (5G) [15] where the communication speed is of utmost importance.

It has been shown in [7] that the unfolded Tanner graph of the BP algorithm naturally becomes a partially connected NN, which enables the use of DL techniques on top of the BP decoding algorithm. Thus, a great deal of research mainly focuses on designing DL algorithms for BP decoding. It was further shown in [7] that learn-able weights can be assigned to the unfolded Tanner graph of BP decoder of high-density parity-check (HDPC) codes to enable not only error correction performance improvement, but also latency reduction over the original BP decoder. In [8], [9], performance degradation due to the min-sum approximation was recovered by assigning learn-able multiplicative or additive parameters in BP decoding of short Bose-Chaudhuri-Hocquenghem (BCH) codes. A similar technique is applied in [10] for the case of polar codes. However, all the aforementioned algorithms use DL as a refinement technique as the decoding process is still carried out by BP decoding.

An alternative approach is to directly train a NN model to decode a codeword. It has been observed in [11], [12] that due to the exponential complexity of the learning space [13], NN-based decoders can only achieve MAP performance for short code lengths. For long code lengths, one solution is to make use of partitioned NN (PNN) decoders as presented in [14]. However, BP decoding often requires a high number of iterations to obtain equivalent decoding performance as that of SC decoding [6]. Therefore, the PNN decoder requires high number of BP iterations to achieve a reasonable error correction performance which in turn reduces the speed of decoding.

In this paper, we propose an NN-based neural SC (NSC) decoder that consists of multiple constituent NN decoders which are connected together using SC decoding. We train the NN decoders with the internal log-likelihood ratio (LLR) values given by SC decoding and show that the proposed approach introduces no latency overhead which are inherently caused by BP iterations. We further show that with the SC coupling stage, the proposed NSC decoder has significantly smaller decoding latency while maintaining the same error correction performance in comparison with the PNN decoder of [14].

The remainder of this paper is organized as follows.

978-1-5386-3512-4/18/$31.00 ©2018 IEEE

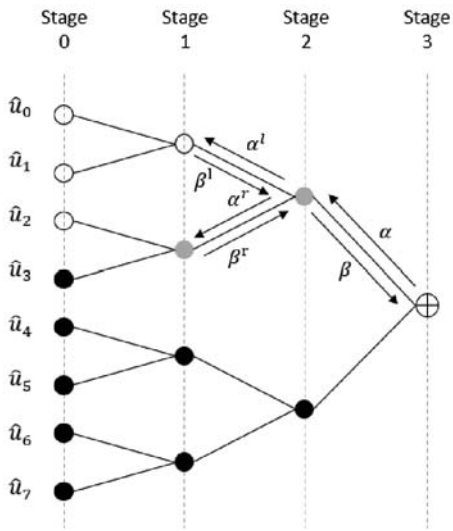

Fig. 1: SC decoding on a binary tree for $\mathcal{P}(8,5)$ and $\{u_0, u_1, u_2\} \in \mathcal{F}$.

Section II briefly introduces polar codes and the SC and NN decoding algorithms. In Section III, the proposed NSC decoder is introduced and the experimental results are provided in Section IV. Finally, Section V draws the main conclusions of the paper.

II. Preliminaries

A. Polar Codes

A polar code $\mathcal{P}(N,K)$ of length N with K information bits is constructed by applying a liner 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 product of the polarizing matrix $\boldsymbol{G} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, and $n = \log_2 N$. The vector $\boldsymbol{u}$ contains a set $\mathcal{A}$ of K information bits and a set $\mathcal{F}$ of $N-K$ frozen bits. The positions and the value of the frozen bits are known to the encoder and the decoder. The codeword $\boldsymbol{x}$ is then modulated and sent through the channel. In this paper, binary phase-shift keying (BPSK) modulation and additive white Gaussian noise (AWGN) channel model are considered.

B. Successive-Cancellation Decoding

Fig. 1 illustrates the binary tree representation of a polar code $\mathcal{P}(8,5)$ and its corresponding SC decoding. For a node of length N_ν, the soft LLR values $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \ldots, \alpha_{N_\nu-1}\}$ traverse from parent to child nodes, while the estimated hard values $\boldsymbol{\beta} = \{\beta_0, \beta_1, \ldots, \beta_{N_\nu-1}\}$ pass through the reverse direction. The LLR vectors $\boldsymbol{\alpha}^{\mathrm{l}} = \{\alpha^{\mathrm{l}}_0, \alpha^{\mathrm{l}}_1, \ldots, \alpha^{\mathrm{l}}_{\frac{N_\nu}{2}-1}\}$ and $\boldsymbol{\alpha}^{\mathrm{r}} = \{\alpha^{\mathrm{r}}_0, \alpha^{\mathrm{r}}_1, \ldots, \alpha^{\mathrm{r}}_{\frac{N_\nu}{2}-1}\}$ of the left and right-child nodes can be computed as

$$\alpha^{\mathrm{l}}_i = 2\operatorname{arctanh}\left(\tanh\left(\frac{\alpha_i}{2}\right)\tanh\left(\frac{\alpha_{i+\frac{N_\nu}{2}}}{2}\right)\right), \quad (1)$$

$$\alpha^{\mathrm{r}}_i = \alpha_{i+\frac{N_\nu}{2}} + (1-2\beta^{\mathrm{l}}_i)\alpha_i, \quad (2)$$

whereas the estimated hard values $\boldsymbol{\beta}$ of the parent node are updated from those of the left and right-child nodes $\boldsymbol{\beta}^{\mathrm{l}} = \{\beta^{\mathrm{l}}_0, \beta^{\mathrm{l}}_1, \ldots, \beta^{\mathrm{l}}_{\frac{N_\nu}{2}-1}\}$ and $\boldsymbol{\beta}^{\mathrm{r}} = \{\beta^{\mathrm{r}}_0, \beta^{\mathrm{r}}_1, \ldots, \beta^{\mathrm{r}}_{\frac{N_\nu}{2}-1}\}$ as

$$\beta_i = \begin{cases} \beta^{\mathrm{l}}_i \oplus \beta^{\mathrm{r}}_i, & \text{if} \quad i < \frac{N_\nu}{2}, \\ \beta^{\mathrm{r}}_{i-\frac{N_\nu}{2}}, & \text{otherwise}, \end{cases} \quad (3)$$

where $\oplus$ is the bitwise XOR operation. At the leaf level, the i-th estimated message bit $\hat{u}_i$ can be obtained as

$$\hat{u}_i = \begin{cases} 0, & \text{if } i \in \mathcal{F} \text{ or } \alpha_i \geq 0, \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

Furthermore, (1) can be approximated using the hardware-friendly version proposed in [16]:

$$\alpha^{\mathrm{l}}_i = \operatorname{sgn}(\alpha_i)\operatorname{sgn}(\alpha_{i+\frac{N_\nu}{2}})\min(|\alpha_i|, |\alpha_{i+\frac{N_\nu}{2}}|). \quad (5)$$

It was shown in [1] that the latency of SC decoding algorithm can be represented in terms of the number of time steps as

$$\mathcal{T}_{\mathrm{SC}} = 2N - 2. \quad (6)$$

C. Neural Network Decoding

A fully connected NN [17] decoder contains T hidden layers with the size of $\{L_1, L_2, \ldots, L_T\}$, where $L_t > 0$ and $1 \leq t \leq T$. The size of the input and output layers is equal to the number of input LLR values or the number of output bits, denoted as N_ν. The full network structure can then be expressed as $\{L_0, L_1, \ldots, L_T, L_{T+1}\} = \{N_\nu, L_1, \ldots, L_T, N_\nu\}$, where L_0 and L_{T+1} denote the size of the input and output layers, respectively. Each layer t for $1 \leq t \leq T+1$ has an associated $L_{t-1} \times L_t$ weight matrix $\boldsymbol{w}_t$ and a $1 \times L_t$ bias vector $\boldsymbol{b}_t$. The output vector of layer t is denoted by $\boldsymbol{o}_t$ and is computed as

$$\boldsymbol{o}_t = f_t(\boldsymbol{o}_{t-1}) = \phi_t(\boldsymbol{o}_{t-1}\boldsymbol{w}_t + \boldsymbol{b}_t), \quad (7)$$

where $\boldsymbol{o}_0 = \boldsymbol{\alpha}$ and ϕ_t is a non-linear activation function such that

$$\phi_t(\boldsymbol{y}) = \begin{cases} \max(0, \boldsymbol{y}), & \text{if } 1 \leq t \leq T, \\ \frac{1}{1+e^{-\boldsymbol{y}}}, & \text{if } t = T+1. \end{cases} \quad (8)$$

Note that the *sigmoid* function $\frac{1}{1+e^{-\boldsymbol{y}}}$ [17] constrains the outputs of the last layer within the range of $(0,1)$, which are also understood as the probability of the output bits to be 1. Let us consider $\hat{\boldsymbol{u}} = \{\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N_\nu-1}\}$ represents the output bits and $\boldsymbol{P} = \{P_0, P_1, \ldots, P_{N_{\nu-1}}\}$ represents the vector of probabilities such that for $0 \leq i < N_\nu - 1$, $P_i = Pr(\hat{u}_i = 1)$. The NN decoder with the given layer configuration can be written as

$$\boldsymbol{P} = f_{\mathrm{NN}}(\boldsymbol{\alpha}) = f_{T+1}(f_T(\ldots(f_1(\boldsymbol{\alpha})))), \quad (9)$$

where $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \ldots, \alpha_{N_\nu-1}\}$ is the vector of the input LLR values. Once the probability of each output bit is obtained, the decoded bits are estimated as

$$\hat{u}_i = \begin{cases} 0, & \text{if } i \in \mathcal{F} \text{ or } P_i \leq 0.5, \\ 1, & \text{otherwise.} \end{cases} \quad (10)$$

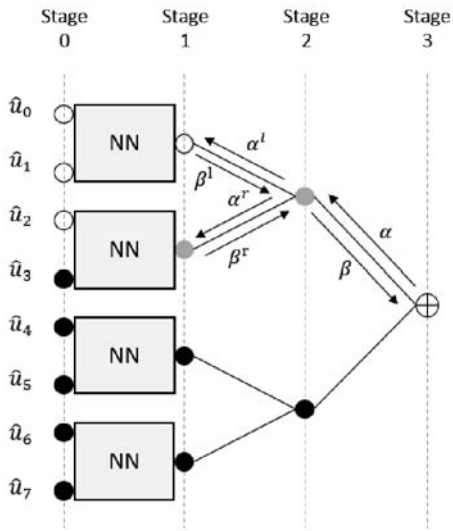

Fig. 2: NSC decoding with NN decoders applied at stage 1 of $\mathcal{P}(8,5)$ with $\{u_0, u_1, u_2\} \in \mathcal{F}$.

Algorithm 1: Collect training data for NN decoders

Input : $\boldsymbol{u}, \boldsymbol{\alpha}$

Output: $\boldsymbol{u}_k$=$\{u_{k2^s}, u_{k2^s+1}, \ldots, u_{(k+1)2^s-1}\}$, $\boldsymbol{\alpha}_k^s$=$\{\alpha_{k2^s}^s, \alpha_{k2^s+1}^s, \ldots, \alpha_{(k+1)2^s-1}^s\}$

/* Hard values calculation */

$\boldsymbol{\beta}^0 = \boldsymbol{u}$;

for $i \leftarrow 1$ **to** $n-1$ **do**

  Calculate $\boldsymbol{\beta}^i$ *from* $\boldsymbol{\beta}^{i-1}$;

end

/* Internal LLR values calculation */

$\boldsymbol{\alpha}^n = \boldsymbol{\alpha}$;

for $i \leftarrow n-1$ **to** s **do**

  Calculate $\boldsymbol{\alpha}^i$ *from* $(\boldsymbol{\alpha}^{i+1}, \boldsymbol{\beta}^i)$;

end

/* Store the training data for each NN decoder */

for $k \leftarrow 0$ **to** $N/2^s - 1$ **do**

  Store $\boldsymbol{u}_k$ *and* $\boldsymbol{\alpha}_k^s$ *for the* k*-th NN decoder*;

end

The weight matrix $\boldsymbol{w}_t$ and the bias vector $\boldsymbol{b}_t$ are trained using backpropagation, and the loss function can either be the mean-squared-error (MSE) or the binary-cross-entropy (BCE) functions [17] as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{M} \sum_{j=0}^{M-1} \sum_{i=0}^{N_\nu - 1} (u_i^{(j)} - P_i^{(j)})^2, \tag{11}$$

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{M} \sum_{j=0}^{M-1} \sum_{i=0}^{N_\nu - 1} \Big[(1 - u_i^{(j)}) \ln (1 - P_i^{(j)}) + u_i^{(j)} \ln P_i^{(j)} \Big], \tag{12}$$

where $u_i^{(j)}$ and $P_i^{(j)}$ are the bit value and the output probability of the i-th bit in the j-th message word of the mini batch, respectively. Each message word contains N_ν bits, while the mini batch has a size of M.

The number of time steps required to finish the decoding process in the NN decoder is dependent on the number of hidden layers and can be calculated as $\mathcal{T}_{\text{NN}} = T + 1$. In [14], a PNN decoder was introduced based on the partitioning technique of [5] to reduce the complexity of the learning space. For a PNN decoder with partitions of length 2^s $(0 \leq s \leq n)$, the number of required decoding time steps is

$$\mathcal{T}_{\text{PNN}} = \frac{N}{2^s}(T+1) + 2\frac{N}{2^s} \log_2 \frac{N}{2^s}. \tag{13}$$

III. Neural Successive Cancellation Decoding

The structure of the proposed NSC decoder is determined by first selecting a stage s in the SC decoding tree and replacing each constituent SC decoder of length 2^s by an NN decoder. Fig. 2 illustrates the proposed NSC decoder where the replacement is carried out at stage 1, thus each NN decoder receives two internal LLR values and predicts two output bits. Unlike other fast SC decoders that need to be manually designed to decode special constituent codes [18], the NN decoder in this paper is trained to decode any node without considering any specific patterns formed by frozen bit and information bit locations.

The process of collecting training data for each NN decoder is summarized in Algorithm 1. The message word and the channel LLR values are denoted as $\boldsymbol{u}$ and $\boldsymbol{\alpha}$, respectively. Let i and k denote the stage index and the NN decoder index, respectively, where $0 \leq i \leq n-1$ and $0 \leq k \leq N/2^s - 1$. It should be noted that each NN is of size 2^s. The vector of internal LLR and hard values at stage i are denoted as $\boldsymbol{\alpha}^i = \{\alpha_0^i, \alpha_1^i, \ldots, \alpha_{N-1}^i\}$ and $\boldsymbol{\beta}^i = \{\beta_0^i, \beta_1^i, \ldots, \beta_{N-1}^i\}$, respectively. The training data of each NN decoder is obtained by assuming that the SC decoder has perfect knowledge of the transmitted bits. Thus, the hard values of all stages are calculated using the message word and the internal LLR values from stage n to stage s are calculated given that all the previous bits are decoded correctly. Finally, the internal LLR values at stage s, $\boldsymbol{\alpha}_k^s = \{\alpha_{k2^s}^s, \alpha_{k2^s+1}^s, \ldots, \alpha_{(k+1)2^s-1}^s\}$, and the corresponding message word, $\boldsymbol{u}_k = \{u_{k2^s}, u_{k2^s+1}, \ldots, u_{(k+1)2^s-1}\}$, of the k-th NN decoder is stored.

After the training phase, each constituent NN decoder obtains its weight and bias matrices and is ready for decoding. The decoding process of the proposed NSC decoding algorithm is summarized in Algorithm 2. The NSC decoding algorithm makes use of SC decoding to generate internal LLR values for each NN decoder. At stage s, the k-th NN decoder infers the input LLR values $\boldsymbol{\alpha}_k^s$ using its trained weights and biases. The output bits of this NN decoder, denoted as $\hat{\boldsymbol{u}}_k$, are then estimated in accordance with (10). Then, $\hat{\boldsymbol{\beta}}_k^s$ at stage s, given by this NN decoder, is updated. The decoding process is continued until the last NN decoder outputs its estimation.

Let us consider the k-th NN decoder at stage s. The output

Algorithm 2: NSC Decoding

Input : $\boldsymbol{\alpha}$
Output: $\hat{\boldsymbol{u}}$
/* Successively apply NN decoding for each constituent code */
for $k \leftarrow 0$ **to** $N/2^s - 1$ **do**
  Calculate $\boldsymbol{\alpha}_k^s$ *using SC decoding*;
  Estimate $\hat{\boldsymbol{u}}_k$ *from* $\boldsymbol{\alpha}_k^s$ *using NN decoding*;
  Update $\hat{\boldsymbol{\beta}}_k^s$ *from* $\hat{\boldsymbol{u}}_k$;
end
return $\hat{\boldsymbol{u}}$

estimation of this NN decoder in accordance with (9) is:

$$\begin{aligned} \boldsymbol{P} &= f_{\text{NN}}(\boldsymbol{\alpha}_k^s) \\ &= f_{\text{NN}}(f_{\text{SC}}(\boldsymbol{\alpha}, \hat{\boldsymbol{u}}_0, \hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_{k-1})). \end{aligned} \tag{14}$$

The function f_{SC} implies SC decoding with the use of (2)-(5), given the channel LLR values, $\boldsymbol{\alpha}$, and the hard values, which are calculated from the output bit vectors $\hat{\boldsymbol{u}}_m$ of all NN decoders with $0 \leq m \leq k-1$. It can be observed from (14) that the SC decoding used in the proposed decoder (similar to the BP decoding in [14]) acts like a feature extractor that maps the feature space $\boldsymbol{\alpha} \in \mathbb{R}^N$ to a lower-dimensional feature space $\boldsymbol{\alpha}_k^s \in \mathbb{R}^{2^s}$.

It is worth mentioning that for sufficient training, each partitioned NN decoder needs to be trained from the dataset $(\boldsymbol{u}_k, \boldsymbol{\alpha}_k^s)$ that is generated from the channel LLR values with all the possible values of information bits. Therefore, the size of the training data for each constituent NN decoder has a space complexity of $\Theta(2^K)$ due to the fact that

$$\boldsymbol{\alpha} = \mathbf{1} - 2\boldsymbol{u}\boldsymbol{G}^{\otimes n} + \boldsymbol{z}, \tag{15}$$

where $\boldsymbol{z}$ is the AWGN channel noise, $\mathbf{1}$ is the all-one vector of length N, and $\boldsymbol{u}$ contains K information bits. However, the SC coupling stage simplifies the learning problem of the individual NN decoders by reducing the feature size.

The decoding latency in terms of the number of time steps for the proposed NSC decoder can be calculated as

$$\mathcal{T}_{\text{NSC}} = \frac{N}{2^s}(T+1) + 2\frac{N}{2^s} - 2, \tag{16}$$

which is always smaller than the latency of the PNN decoder as given in (13), if the two decoders have the same number of NN decoders and $s < n$. This is illustrated in Fig. 3 for PNN and NSC decoding of a polar code of length $N = 128$, when $T = 3$, and for different values of the stage index s. It can be seen that as the number of partitions increases, the saving in the number of time steps achieved by NSC is more significant.

IV. Experimental Results

In this section, we examine the error correction performance of the proposed NSC decoder in terms of bit error rate (BER) and frame error rate (FER). To this end, the NSC decoder is applied to $\mathcal{P}(128, 64)$, which is constructed for

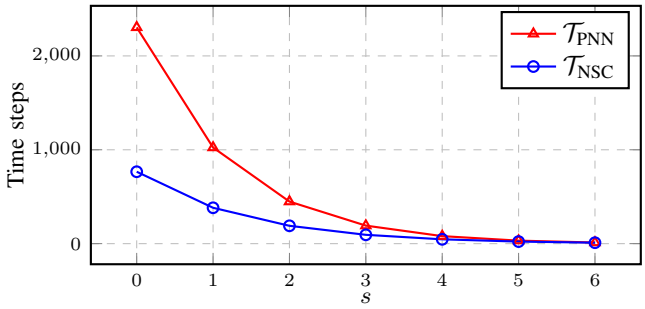

Fig. 3: Decoding latency comparison between NSC and PNN decoders with $N = 128$, $T = 3$, and different values of s.

SNR $= 5$ dB based on [19]. In order to select the value of s at which the NN decoders are present, we first fix the polar code and the constituent NN decoder structure. We then progressively increase the value of s from 0 to $\log_2 N$ and apply the proposed algorithm for each value of s. We pick the largest value of s with which the decoding performance of the proposed algorithm is preserved in comparison with the original SC decoding. We use (12) as the loss function and use stochastic gradient descent to minimize the loss. The training phase is implemented using Keras [20] with TensorFlow back-end [21]. Adam [22] and early-stopping techniques [23] are used as the methods of optimization and regularization, respectively.

In order to obtain the training and validation data, 5000000 random codewords are generated first. Then a random segmentation containing 80% of the generated codewords is used as the training set, and the remaining 20% is used as the validation set. Note that the above data ratio is a practical parameter suggested by [20]. In addition, in order to have a fair comparison, the testing process of the proposed decoder is consistent with other considered decoders, in which at least 10000 codewords are tested and at least 50 frames in error are captured. It should be noted that the batch size in use is 10000 and the maximum number of training epochs is 2000.

The error correction performance of the proposed NSC decoder is compared with the PNN decoder of [14], and the SC and BP decoders, as shown in Fig. 4. For a fair comparison, the PNN decoder is designed to have the same number of partitions as that of the NSC, i.e. 8 partitions. Furthermore, each constituent NN decoder of the NSC and the PNN decoders has the same network configuration with $T = 3$ and $\{L_0, L_1, L_2, L_3, L_4\} = \{16, 512, 256, 128, 16\}$. For the BP decoder, 30 iterations are used to provide the same error correction performance as the SC decoder. It should be noted that the time steps required by BP decoding with I iterations can be calculated as

$$\mathcal{T}_{\text{BP}} = 2I \log_2 N. \tag{17}$$

It can be seen in Fig. 4 that the NSC decoder has similar error correction performance in comparison with PNN, SC, and BP decoders.

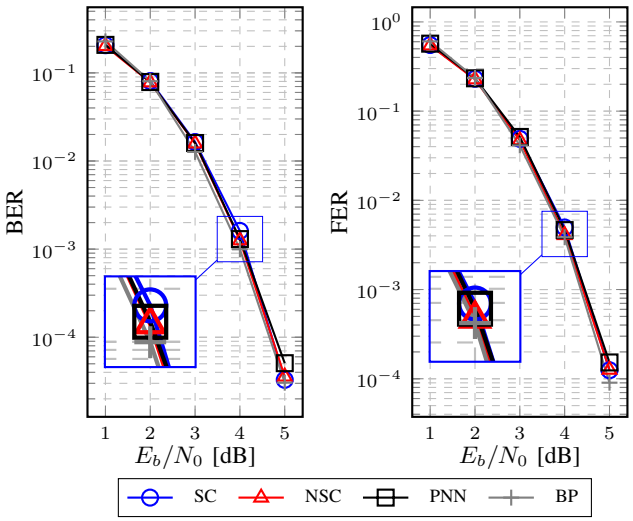

Fig. 4: BER and FER performance comparison of NSC decoding of $\mathcal{P}(128, 64)$, with the PNN decoder of [14], and SC and BP decoders. The NSC and the PNN decoders are implemented with 8 constituent NN decoders and 30 iterations are used in the BP decoder.

TABLE I: Time-step requirements of decoding $\mathcal{P}(128, 64)$ with the proposed NSC decoder in comparison with the PNN decoder of [14], and the SC and BP decoders. The NSC and the PNN decoders are implemented with 8 constituent NN decoders and 30 iterations are used in the BP decoder.

Decoder	SC	BP	PNN	NSC
Latency [Time steps]	254	420	80	46

Table I summarizes the decoding latency in time steps of the proposed NSC decoder for decoding $\mathcal{P}(128, 64)$ and compares it with the latency of PNN, SC, and BP decoders with the same parameters as the decoders in Fig. 4. It can be observed that at the same error correction performance, the NSC decoder requires only 57.5% of the number of time steps required for the PNN decoder. Moreover, the proposed NSC decoder reduces the time step requirements of SC and BP decoders by 81.9% and 89.0%, respectively.

V. Conclusion

In this paper we proposed a neural successive cancellation (NSC) decoder for polar codes based on the partitioning technique of [5]. Unlike the partitioned neural network (PNN) decoder introduced in [14], the constituent neural network (NN) decoders in the proposed NSC decoder are connected together using successive cancellation (SC) decoding. We demonstrated that at the same error correction performance, the decoding latency of the proposed NSC decoder in terms of the number of time steps is up to 89% smaller than that of SC, belief propagation, and PNN decoders. Our future work will investigate the decoding capability of the more powerful NN models, such as recurrent NN (RNN), when applied to the proposed NSC decoder.

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] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, March 2015.
[3] 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, June 2015.
[4] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, August 2017.
[5] S. A. Hashemi, C. Condo, F. Ercan, and W. J. Gross, "Memory-efficient polar decoders," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 7, no. 4, pp. 604–615, October 2017.
[6] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6496–6506, October 2014.
[7] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Annual Allerton Conf. on Commun. Control and Computing*, February 2016, pp. 341–346.
[8] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *IEEE Int Symp. on Inf. Theory*, August 2017, pp. 1361–1365.
[9] 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.
[10] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *IEEE Int. Workshop on Signal Process. Syst.*, November 2017, pp. 1–6.
[11] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *Annual Conf. on Inf. Sciences and Syst.*, May 2017, pp. 1–6.
[12] W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang, "Performance evaluation of channel decoding with deep neural networks," *arXiv preprint arXiv:1711.00727*, 2017.
[13] X.-A. Wang and S. B. Wicker, "An artificial neural net Viterbi decoder," *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 165–171, February 1996.
[14] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *IEEE Global Commun. Conf.*, December 2017, pp. 1–6.
[15] 3GPP, "Final report of 3GPP TSG RAN WG1 #87 v1.0.0." *Reno, NV, USA*, 2016. [Online]. Available: http://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_87/Report/Final_Minutes_report_RAN1%2387_v100.zip
[16] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 289–299, October 2013.
[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, May 2015.
[18] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, April 2014.
[19] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, Oct 2013.
[20] F. Chollet *et al.*, "Keras," 2015.
[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.
[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[23] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, March 1995.