Neural Successive Cancellation Decoding of Polar Codes

Nghia Doan, Seyyed Ali Hashemi, and Warren J. Gross

nghia.doan@mail.mcgill.ca, seyyed.hashemi@mail.mcgill.ca, warren.gross@mcgill.ca

McGill University

McGill

Abstract

- Polar Codes
  - Capacity-achieving for codes of infinite length
  - Low-complexity encoding and decoding
  - Selected for 5G eMBB control channel
- Partitioned Neural-network Decoding [1]
  - Applicable for short polar codes
  - Suitable for latency-critical applications
  - Problem: high decoding latency caused by Belief Propagation (BP) coupling stage
  - Solution: use Successive-cancellation (SC) decoding instead of BP decoding

Polar Codes

- $\mathcal{P}(N, K)$: polar code of length N and rate $\frac{K}{N}$
- K best reliable bits to transmit information bits
- Successive-cancellation (**SC**) Decoding:
  - Mediocre error-correction performance for short codes
  - Latency: $\mathcal{T}_{\text{SC}} = 2N - 2$ (time steps)
- Belief Propagation (**BP**) Decoding:
  - Reasonable error-correction performance with enough iterations
  - Latency: $\mathcal{T}_{\text{BP}} = 2I \log_2 N$ (time steps)

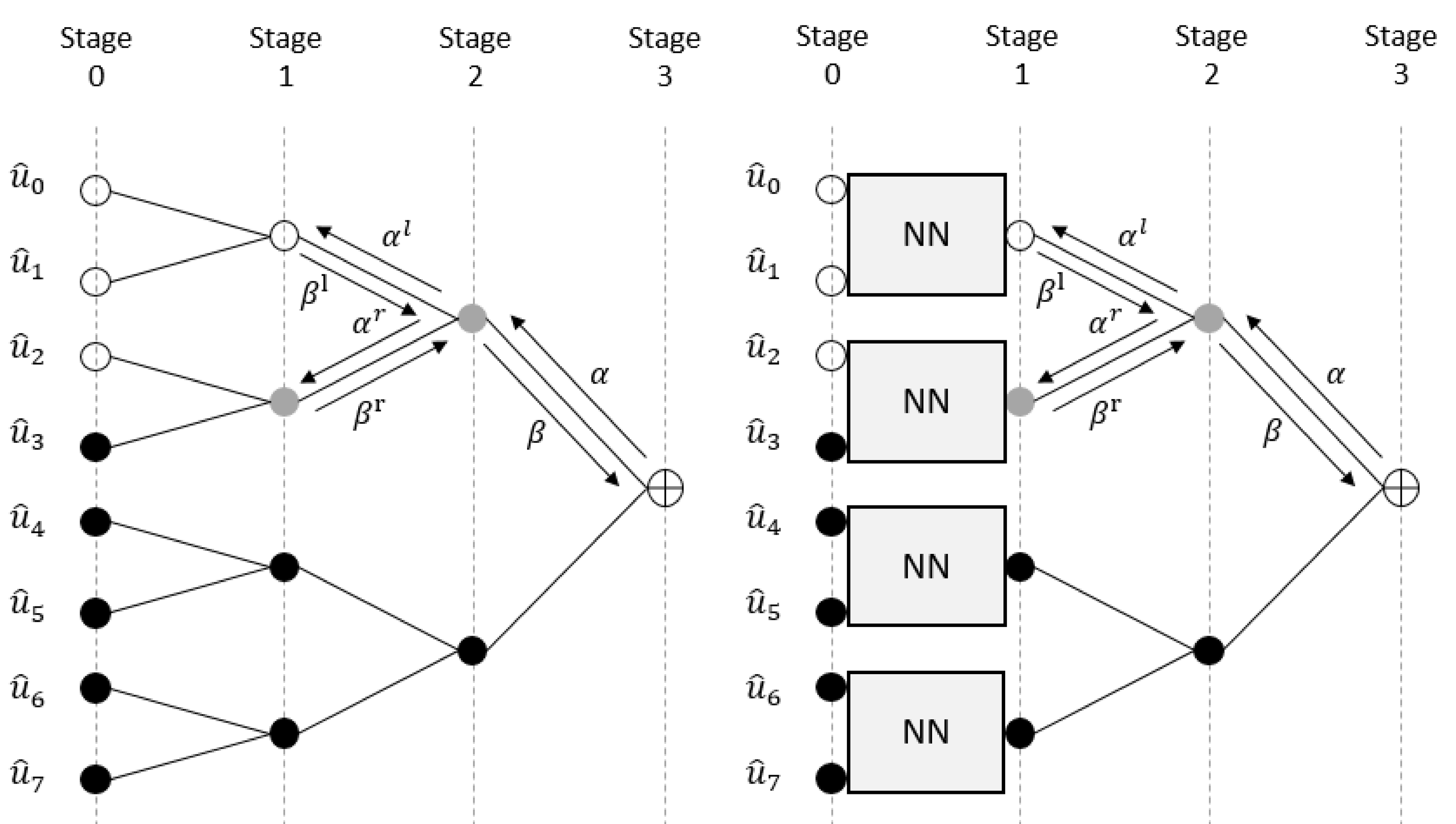
Partitioned Neural Network (PNN) Decoding [1]

- Multiple NN-based decoders are connected using BP decoding
- Has the same decoding performance as SC decoding
- Latency: $\mathcal{T}_{\text{PNN}} = \frac{N}{2^s}(T+1) + 2\frac{N}{2^s}\log_2\frac{N}{2^s}$ (time steps)

  where s: the partition stage, $0 \leq s \leq \log_2 N$

Neural Successive Cancellation (NSC) Decoding

- Training
  - The internal LLRs at stage s are calculated using SC decoding, given the channel LLRs y and the correct message word u
  - Each NN-based decoder is trained with its corresponding partitioned internal LLRs and correct message bits
  - Each NN-based decoder obtains a set of trained weight and bias values
- Decoding
  - The decoding scheduling is similar to that of Partitioned Successive Cancellation List (PSCL) Decoder [2], where each SCL decoder is replaced by a NN-based decoder
  - SC decoding is used to supply soft information for all NN-based decoders
  - The decoding is finished when the last NN-based decoder outputs its estimation
  - Latency: $\mathcal{T}_{\text{NSC}} = \frac{N}{2^s}(T+1) + 2\frac{N}{2^s} - 2$ (time steps)



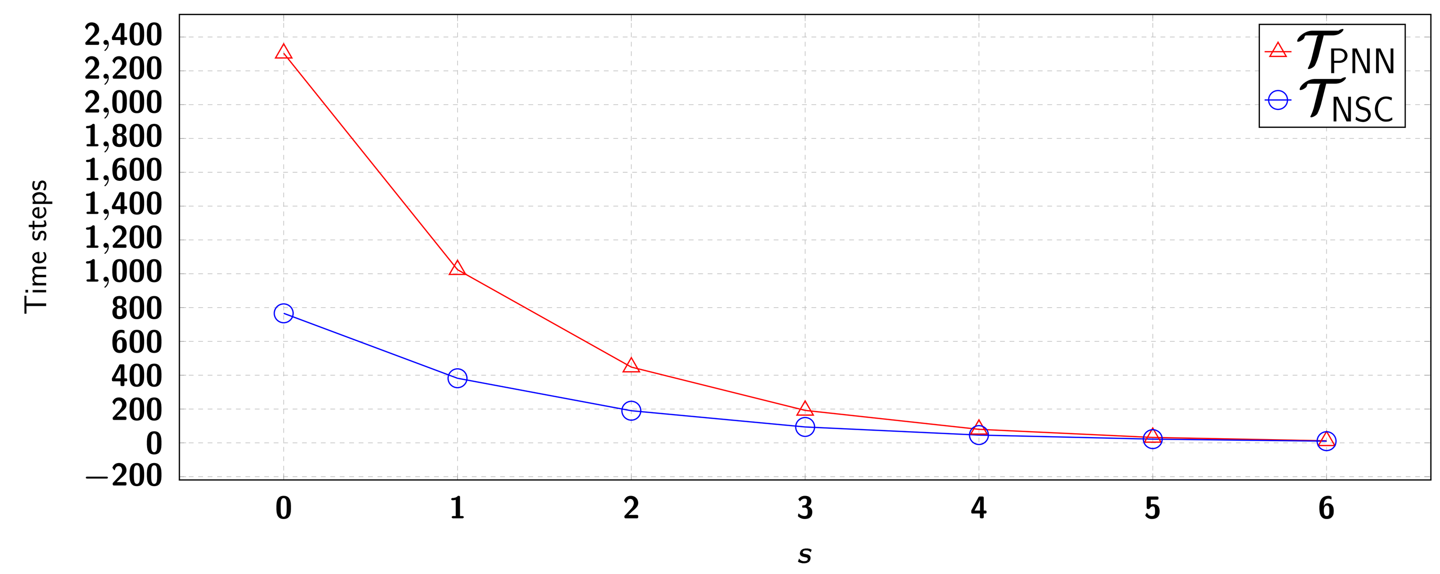
(a) SC decoding of $\mathcal{P}(8, 5)$ (b) NSC decoding of $\mathcal{P}(8, 5)$

Configurations

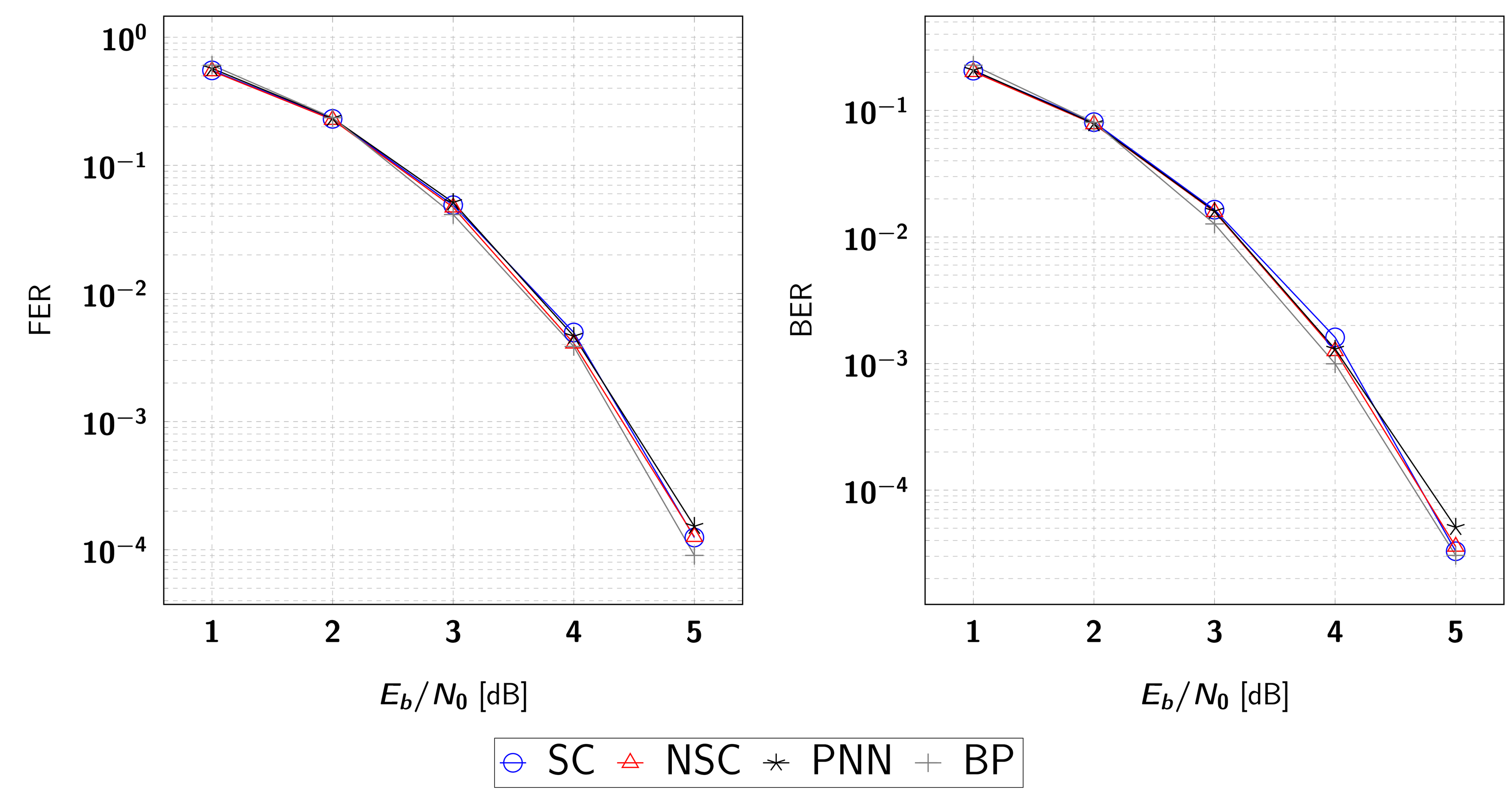
- Examined polar code:
  - $\mathcal{P}(128, 64)$ constructed for $SNR = 5dB$
  - AWGN channel, BPSK modulation
- NSC decoder
  - Partition stage: $s = 4$
  - NN-based decoder network size: $\{16, 512, 256, 128, 16\}$
- Training for constituent NN-based decoders
  - Framework: Keras [3] with TensorFlow [4] back-end
  - Optimization and regularization: Adam [5] and early stopping [6]
  - Training set: 4×10^6 random codewords
  - Validation set: 10^6 random codewords
- Evaluation setup:
  - Comparison: SC, BP, PNN and NSC decoders for the examined polar code
  - Termination condition: at least 10^5 frames and at least 50 error frames

Experimental Results

- PNN and NSC decoding latency with $\mathcal{P}(128, 64)$ and various values of s



- Decoding performance comparison for $\mathcal{P}(128, 64)$ and $s = 4$



- Decoding latency comparison for $\mathcal{P}(128, 64)$ and $s = 4$

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

Conclusion

- We proposed a NSC decoder which uses constituent NN-based decoders and SC decoding
- The proposed decoder has the same decoding performance when compared to PNN, SC, and BP decoders
- The decoding latency of the NSC decoder is **42.5%**, **81.9%**, and **89%** smaller than that of PNN, SC, and BP decoders, respectively.

Reference

[1] 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.

[2] 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.

[3] F. Chollet *et al.*, "Keras," 2015.

[4] 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.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[6] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, March 1995.