

# Decoding Convolutional Hadamard Codes and Turbo Hadamard Codes using Recurrent Neural Networks

Sheng Jiang and Francis C. M. Lau

Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University,  
Hong Kong SAR, China

Emails: s.jiang@polyu.edu.hk, francis-cm.lau@polyu.edu.hk

**Abstract**—In this paper, a Recurrent Neural Network (RNN) based decoder is proposed for the decoding of convolutional Hadamard codes (CHC) and Turbo Hadamard Codes (THC). Moreover, a long short-term memory (LSTM) network is adopted to realize the RNN decoder, forming the LSTM-CHC decoder and LSTM-THC decoder. Also, the proposed LSTM-THC decoder consists of several serially concatenated LSTM-CHC decoders, which are pre-trained separately. The end-to-end LSTM-THC decoder is then trained based on the pre-trained weights. Simulations are performed on the LSTM-CHC/LSTM-THC decoders and their error performances are compared with those of the conventional decoders.

**Index Terms**—convolutional Hadamard code, turbo Hadamard code, Recurrent Neural Networks.

## I. INTRODUCTION

Deep learning (DL) has been providing new approaches to decoding channel codes. In [1] and [2], Dense Neural Network (DNN) is used to mimic the Belief Propagation (BP) decoders of Bose-Chaudhuri-Hocquenghem (BCH) codes and High-Density Parity-Check (HDPC) codes. Neural BP decoder has also been proposed to decode polar codes in [3] and [4]. On the other hand, Recurrent Neural Network (RNN) is used to mimic the Bahl-Cocke-Jelinek-Raviv (BCJR) decoders of convolutional codes and turbo codes in [5]. However, no deep learning decoder has been investigated to decode channel codes that approaches the ultimate Shannon limit. Turbo Hadamard codes (THC) [6], concatenated zigzag Hadamard codes (CZHC) [7] and Low-Density Parity-Check Hadamard codes (LHC) [8] are a group of ultimate-Shannon-limit approaching codes formed by concatenating Hadamard codes with other codes. In this paper, we propose to decode convolutional Hadamard code (CHC) and THC using RNN. The method could also be applied to CZHC as the code structure and decoding algorithm of CZHC and THC are similar.

We organize the paper as follows. Sect. II reviews the encoder/decoder of CHC and THC. Sect. III describes the structures of our proposed RNN decoder for CHC/THC. Sect. IV shows the details of the training of the RNN decoders and the simulated error performance. Sect. V provides some concluding remarks.

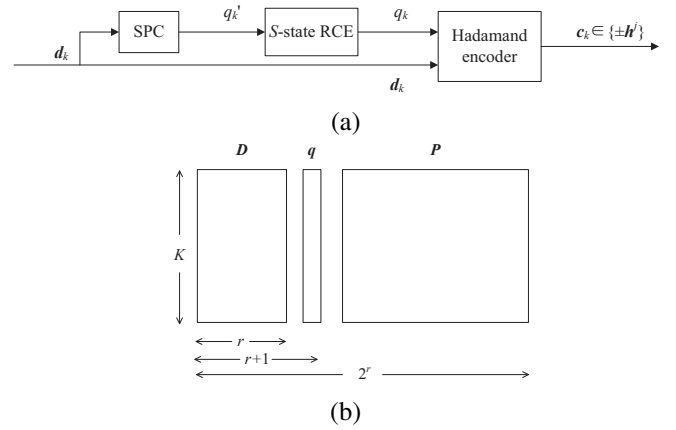


Fig. 1. Convolutional-Hadamard code. (a) Encoder block diagram and (b) code structure. SPC: single-parity check; RCE: recursive convolutional encoder.

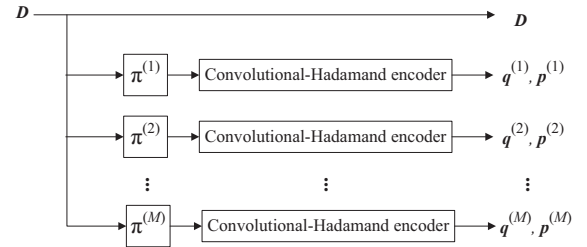


Fig. 2. Structure of turbo-Hadamard code.

## II. REVIEW OF TURBO HADAMARD CODES

An order  $r$  Hadamard matrix is recursively constructed from an order  $r - 1$  Hadamard matrix, i.e.,

$$\mathbf{H}_r = \begin{bmatrix} +\mathbf{H}_{r-1} & +\mathbf{H}_{r-1} \\ +\mathbf{H}_{r-1} & -\mathbf{H}_{r-1} \end{bmatrix} \quad (1)$$

with  $\mathbf{H}_0 = [+1]$ . The codeword set of an order- $r$  Hadamard code is formed by the columns of  $\pm\mathbf{H}_r$ . In an order- $r$  Hadamard code, each codeword has a length of  $2^r$ . Moreover, the  $r$  bits with indices  $\{0, 1, 2, 4, 8, \dots, 2^{r-1}\}$  are denoted as the information bits while the remaining bits are denoted as the parity-check bits.

Fig. 1 shows the encoder block diagram and code structure of a CHC [9]. For an information block  $\mathbf{D}$  of size  $K \times r$ ,

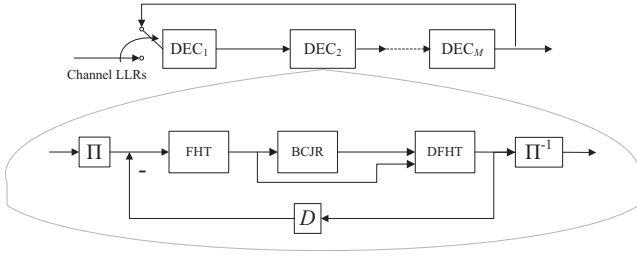


Fig. 3. Block diagram of a turbo Hadamard decoder that consists of  $M$  CHC decoders (sub-decoders). Each CHC decoder further consists of an interleaver, a FHT block, a BCJR block, a DFHT block and a de-interleaver.

we apply an 2-state rate-1/2 recursive-convolutional-encoder (RCE) to generate  $\mathbf{q}$  of size  $K \times 1$  and an order- $r$  Hadamard encoder to generate the Hadamard parity-bits  $\mathbf{P}$  of size  $K \times (2^r - r - 1)$ . Fig. 2 shows the encoder structure of a THC [6], [9], which is the combination of a number of  $M$  CHC derived from the same but interleaved message bits. In particular,  $\pi^{(i)}$  represents the  $i$ -th interleaver; and  $\mathbf{q}^{(i)}$  and  $\mathbf{p}^{(i)}$ , respectively, represent the parity bits generated by the RCE and Hadamard encoders in the  $i$ -th CHC encoder.

The decoding of THC follows a similar principle as turbo code. Referring to Fig. 3, a THC decoder with  $M$  component codes consists of  $M$  serially connected CHC decoders (sub-decoders), where the last sub-decoder is connected to the first sub-decoder. Extrinsic information is exchanged between CHC decoders to enhance the decoder performance. Each of the CHC decoders consists of three blocks: the fast-Hadamard-transform (FHT) block, the Bahl-Cocke-Jelinek-Raviv (BCJR) decoder and the dual-fast-Hadamard-transform (DFHT) block. The CHC decoders compute the *a-posteriori-probability* (APP) likelihood ratio (LLR) and the extrinsic information of the information bits, and then send them to the next CHC decoder. One iteration is completed when the last CHC decoder outputs the LLRs and extrinsic information. Here we denote the number of iterations by  $I$ .

### III. PROPOSED RNN DECODER FOR CHC/THC

The most complex part of the conventional THC decoder is the BCJR decoder. As the BCJR decoder adopts the forward-backward computational algorithm, it introduces a high latency to the decoding process. On the other hand, the forward-backward computational algorithm fits into the natural principle of RNN. In this paper, we investigate the use of a special type of RNN, namely long short-term memory (LSTM), to realize the decoder. This is because LSTM deals with the vanishing gradient problem [10] much better than the traditional RNN. As its name suggests, LSTM not only carries the “short-term memories” of information bits at its decoding stage, but also the “long-term memories” of information bits stored from the previous decoding stages. Moreover, a bi-directional LSTM can even carry the “long-term memories” of the next decoding stages from the backward layers. In other words, the LSTM networks can perfectly imitate the BCJR decoding process. We first investigate the LSTM decoder for CHC. Based on the LSTM-CHC decoders, we construct an end-to-end LSTM-THC decoder.

#### A. CHC decoder based on bi-directional LSTM

In the CHC decoder (sub-decoder) shown in Fig. 3, the FHT block prepares the trellis probabilities from the input *a-priori* (AP) LLRs. The prepared information are then sent to the BCJR decoder to decide which trellis (i.e., the related codeword) has the largest probability. Finally all the information collected from both FHT block and the BCJR decoder are sent to DFHT block to calculate the final APP LLR. In our proposed bi-directional LSTM decoder, the FHT/DFHT blocks are no longer needed. The LSTM networks are capable of merging the preparation stage, the forward/backward stage and the APP calculation stage into one network.

As shown in Fig. 1(b), we assume a CHC segmented into  $K$  (horizontal) blocks with  $r$  information bits and  $2^r - r$  parity-check bits per block. Each block of CHC is actually a Hadamard code of length  $2^r$ . A bi-directional LSTM decoder for the CHC is illustrated in Fig. 4, where the following steps are performed.

- 1) Send the  $K$  blocks of CHC AP LLRs to both the forward layer and the backward layer of the LSTM decoder consecutively. The input length is  $2^r$ . (Note that we can also add an FHT block before feeding the LLRs to the LSTM. However, we find that the decoder performance remains the same.)
- 2) Perform both the forward/backward recursions.
- 3) Collect data from both the forward/backward layers and do batch normalizations to avoid gradient exploding.
- 4) Repeat step 1 to step 3 to get better decoding performance.
- 5) Use a fully-connected dense layer to generate the final APP LLR.

#### B. End-to-end THC decoder based on bi-directional LSTM

An end-to-end LSTM-THC decoder consists of a number of LSTM-CHC decoders plus interleavers and simple logic blocks. For example, the structure of an end-to-end LSTM-THC decoder with  $M = 3$  is shown in Fig. 5. Note that the internal structures of the LSTM-CHC decoders are already illustrated in Fig. 4. Details of the intra-LSTM-CHCs decoding process are described as below.

- 1) We consider the  $i$ -th sub-decoder. The extrinsic LLRs computed during the last iteration of the  $i$ -th LSTM-CHC (which are stored in memory) are first subtracted from the output LLRs of the  $(i - 1)$ -th LSTM-CHC to generate the AP LLRs. These AP LLRs and the parity-check LLRs of the  $i$ -th CHC are then sent to the  $i$ -th LSTM-CHC decoder. Note that the subtraction logics are fixed and no training is needed.
- 2) The  $i$ -th LSTM-CHC decoder produces the APP LLRs. The extrinsic LLRs are computed by subtracting the input AP LLRs from the output APP LLRs, and are stored in memory.
- 3) The APP LLRs are interleaved by an interleaver and then sent to the  $(i + 1)$ -th LSTM-CHC. Note that the interleaver logics are also fixed and no training is needed.

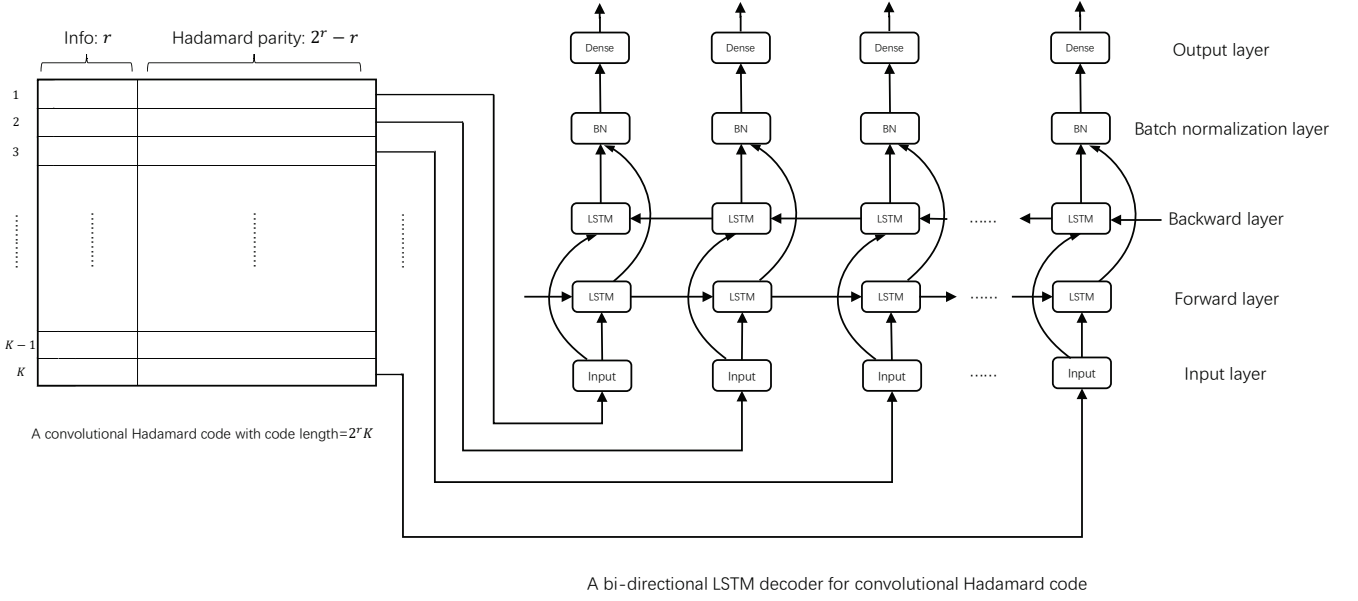
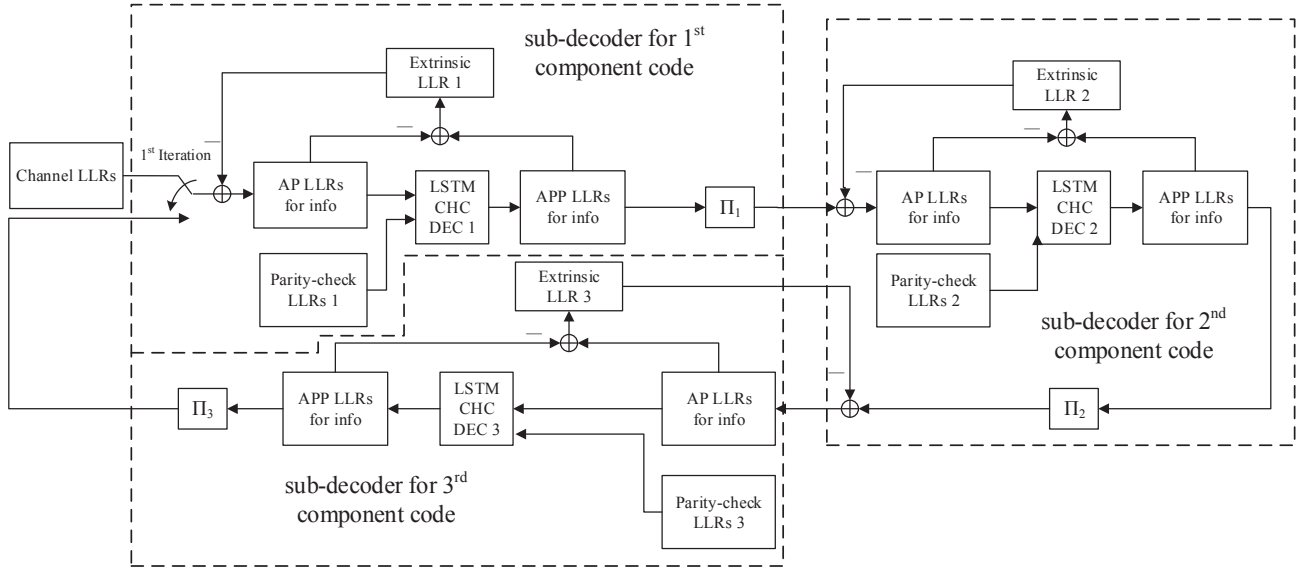


Fig. 4. Illustration of a bi-directional LSTM-CHC decoder.

Fig. 5. End-to-end LSTM-THC decoder structure with  $M = 3$ .

- 4) Iterations of the sub-decoders continue until the maximum number of iterations is achieved.

#### IV. TRAINING AND PERFORMANCE EVALUATION

##### A. LSTM-CHC decoder

In this paper, we consider LSTM decoders for CHC with parameters  $r = 4$ ,  $K = 100$ ; and THC with parameters  $r = 4$ ,  $K = 100$ ,  $M = 3$ . For the LSTM-CHC decoder, 20000 codewords of CHC are generated. The 20000 codes are sent through an additive-white-Gaussian-noise (AWGN) channel with different bit-energy-to-noise-power-spectral-density ratio ( $E_b/N_0$ ) for training. The model structure of the LSTM-CHC is illustrated in Fig. 6. The LSTM-CHC decoder consists of

6 layers. The shape of the input is  $(B, 100, 16)$  where  $B$  denotes the batch size; the number of LSTMs (which equals the number of blocks  $K$  in CHC) is 100; and the input length is  $2^r = 16$ . The number of hidden units in the LSTMs are set to 200 so the shapes of LSTM layers and batch normalization layers are  $(B, 100, 200)$ . The output layer outputs a shape of  $(B, 100, r)$  where the APP LLRs of  $r = 4$  information bits per block is given. Finally, hard decisions are made to the APP LLRs to obtain the decoded bits.

In this study, we train two separate LSTM-CHC decoders with CHCs at  $E_b/N_0 = 2$  dB and  $E_b/N_0 = 10$  dB, respectively. An adaptive moment estimation (ADAM) optimizer with learning rate 0.001 and mean squared error (MSE) loss

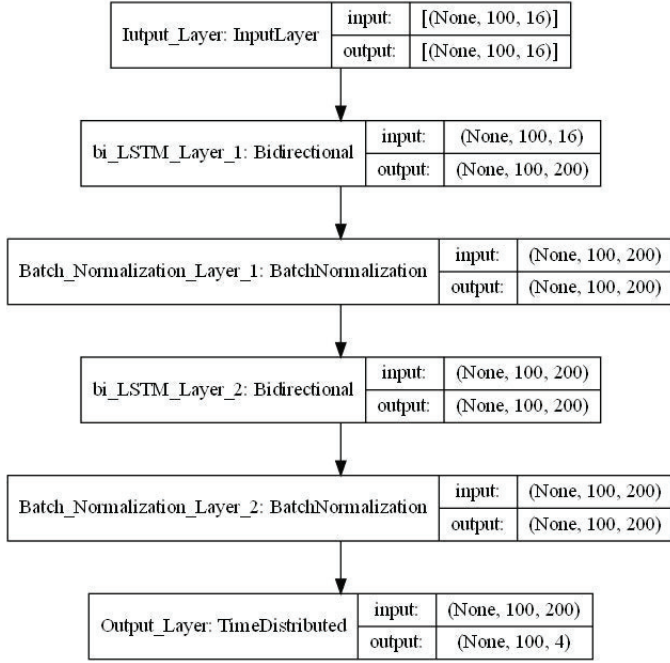


Fig. 6. An example of bi-directional LSTM-CHC decoder architecture.

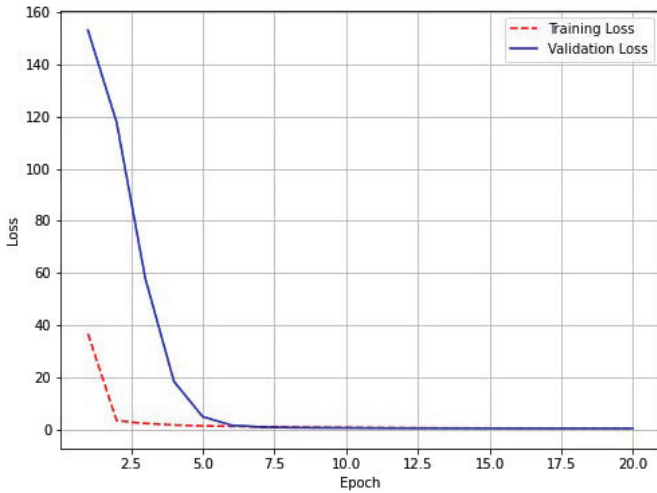


Fig. 7. Training loss and validation loss per epoch in LSTM-CHC.

function are used in the training. CHCs under various  $E_b/N_0$  are used to test the performance of the trained LSTM-CHC decoders. Fig. 7 shows the evolution of the training and validation losses with respect to the number of training epochs when the model is trained at  $E_b/N_0 = 10$  dB. It takes only 7 epochs to make the validation loss close to 0. The BER results compared with conventional CHC decoder are shown in Fig. 8. The LSTM-CHC decoder trained at  $E_b/N_0 = 2$  dB achieves the same performance as the conventional CHC decoder in the low- $E_b/N_0$ -regime, i.e.,  $E_b/N_0 < 6$  dB; but perform badly in the high- $E_b/N_0$ -regime. The LSTM-CHC decoder trained at  $E_b/N_0 = 10$  dB, on the other hand, achieves almost the same performance as the conventional CHC decoder, particularly when  $E_b/N_0 > 9$  dB.

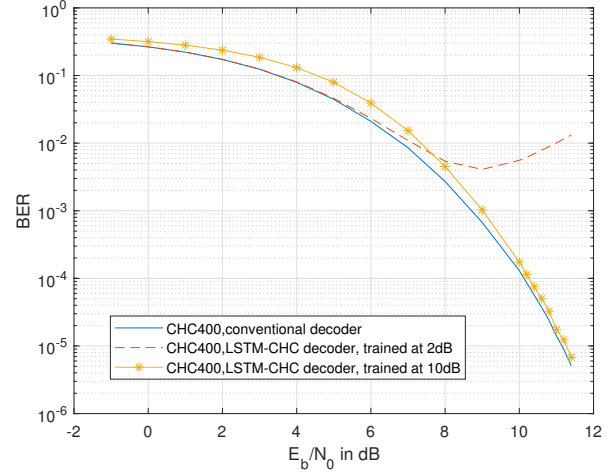


Fig. 8. BER performance of LSTM-CHC decoders.

### B. LSTM-THC decoder

The direct training of the proposed end-to-end LSTM-THC decoder with such complex recurrent networks is challenging because the training from random weights costs a huge amount of time. Instead, we first train the *MI* LSTM-CHC sub-decoders individually and then use the trained weights as the initial weights of the end-to-end LSTM-THC decoders. (Recall that  $I$  denotes the number of iterations of a conventional THC decoder.) Details of the training process are listed as follows.

- 1) Use a THC encoder to randomly generate 20000 THC codewords. These 20000 codes are sent through an AWGN channel under  $E_b/N_0 = -1$  dB. The received noisy observations are then decoded by the standard THC decoder described in [6]. During the decoding process, the intermediate data, i.e., the AP LLRs and the APP LLRs of each sub-decoder, are also recorded.
- 2) Assuming  $I = 4$ , train the  $MI = 12$  LSTM-CHCs using the recorded LLRs with 100 epochs and ADAM optimizer with learning rate 0.001. The mean squared error function is used as the loss function. Record the trained weights of each LSTM-CHC.
- 3) Randomly generate 20000 THC codewords. Train the end-to-end LSTM-THC using the trained weights in Step 2 as initial weights. ADAM optimizer and MSE loss function are used. The learning rate is set to be 0.001 initially and is decreased after a number of epochs. The training is terminated early if the validating loss does not decrease within 10 epochs to prevent overfitting.

The BER performance of LSTM-THC decoder is shown in Fig. 9. It can be observed that the LSTM-THC decoder has a similar performance as the conventional THC decoder at  $E_b/N_0 = -1$  dB, at which the end-to-end LSTM-THC is trained. However, the LSTM-THC decoder performs worse and worse than the conventional decoder as  $E_b/N_0$  increases.

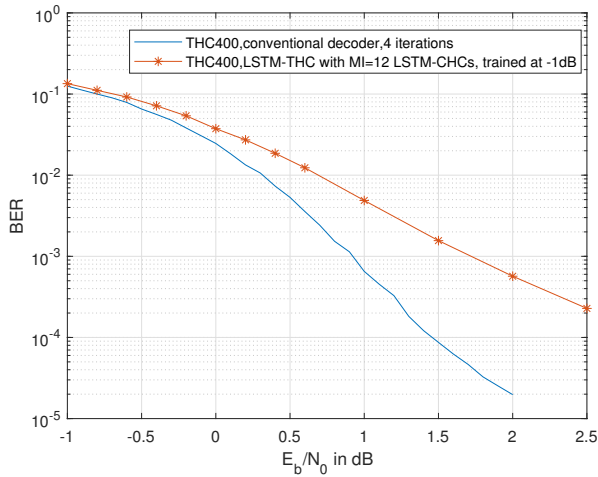


Fig. 9. Comparison of the BER performances of the LSTM-THC decoder and the conventional THC decoder.  $MI = 12$  LSTM-CHCs are used in the LSTM-THC decoder.

## V. CONCLUSION

In this paper, we propose to decode both CHC and THC via LSTM. The BER performance of LSTM-CHC is very close to that of the conventional CHC decoder when the training and testing  $E_b/N_0$  are the same. Since the end-to-end training of LSTM-THC is very time-consuming, we are only able to train the LSTM-THC model at  $E_b/N_0 = -1$  dB. The BER performance of LSTM-THC shows a loss compared to the conventional THC decoder when  $E_b/N_0$  increases from  $E_b/N_0 = -1$  dB. In our future work, we will train the LSTM-THC model at a higher  $E_b/N_0$  and see if the model can produce comparable or better error performance than the conventional THC decoder.

## REFERENCES

- [1] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016, pp. 341–346.
- [2] 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 Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.
- [3] T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink, "On deep learning-based channel decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 2017, pp. 1–6.
- [4] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *GLOBE-COM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.
- [5] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," in *Sixth International Conference on Learning Representations (ICLR)*, 2018.
- [6] L. Ping, W. K. Leung, and K. Y. Wu, "Low-rate turbo-Hadamard codes," *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3213–3224, Dec 2003.
- [7] W. K. R. Leung, G. Yue, L. Ping, and X. Wang, "Concatenated zigzag Hadamard codes," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1711–1723, April 2006.
- [8] G. Yue, L. Ping, and X. Wang, "Generalized low-density parity-check codes based on Hadamard constraints," *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1058–1079, March 2007.

- [9] S. Jiang, P. W. Zhang, F. C. M. Lau, and C.-W. Sham, "An ultimate-Shannon-limit-approaching Gbps throughput encoder/decoder system," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 10, pp. 2169–2173, 2020.
- [10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>



**Sheng Jiang** received the Bachelor of Engineering Degree in Microelectronics from Shanghai Jiaotong University, China; the Master of Engineering Degree in Electronic Engineering from Hong Kong University of Science and Technology, Hong Kong SAR; and the PhD degree from The Hong Kong Polytechnic University, Hong Kong SAR. He is currently a postdoctoral fellow at The Hong Kong Polytechnic University, Hong Kong SAR. His research interests include channel codes, hardware implementation of channel encoder/decoder, and machine learning.



**Francis C. M. Lau** received the BEng(Hons) degree in electrical and electronic engineering and the PhD degree from King's College London, University of London, UK. He is a Professor at the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China. He is also a Fellow of IEEE and a Fellow of IET.

He is a co-author of two research monographs and a co-inventor of six US patents. He has published more than 330 papers. His main research interests include channel coding, cooperative networks, wireless sensor networks, chaos-based digital communications, applications of complex-network theories, and wireless communications. Over the past years, he has secured research grants and consultancy projects from various organizations including the Hong Kong Research Grant Council; Hong Kong Jockey Club; Highways Department, Hong Kong SAR; National Natural Science Foundation of China; and Huawei Technologies Co. Ltd. He is a co-recipient of one Natural Science Award from the Guangdong Provincial Government, China; eight best/outstanding conference paper awards; one technology transfer award; two young scientist awards from International Union of Radio Science; and one FPGA design competition award.

He was the General Co-chair of International Symposium on Turbo Codes & Iterative Information Processing (2018) and the Chair of Technical Committee on Nonlinear Circuits and Systems, IEEE Circuits and Systems Society (2012–13). He served as an associate editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2004–2005 and 2015–2019), IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I (2006–2007), and IEEE CIRCUITS AND SYSTEMS MAGAZINE (2012–2015). He has been a guest associate editor of INTERNATIONAL JOURNAL AND BIFURCATION AND CHAOS since 2010. He also served as a member of the IEEE CAS Society Fellow Evaluation Committee in 2022 and 2023.