Machine Learning for Polar Codes in Small IoT Devices

Xian Liu
Department of Systems Engineering
University of Arkansas at Little Rock
Little Rock, USA

Changcheng Huang
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada

Abstract – **Error-correction in the communication module of IoT systems is a mission-critical task. The Polar code is one of the modern 5G wireless technologies. Most conventional decoding methods are iterative. It is interesting to investigate the feasibility of AI methodology, such as machine learning, for small Polar codes. In this paper, we design a small artificial neural network for this purpose. It is shown that this neural network can successfully decode four small Polar codes: both non-systematic and systematic, with codeword length 8 and 16, respectively.**

Keywords – ***5G communications, artificial neural network, error correction coding, Internet of Things, machine learning.***

I. INTRODUCTION

As a fast-evolving interdisciplinary field, *Internet of Things* (IoT) embraces many cutting-edge concepts arising from the *fifth-generation* (5G) wireless technologies, robotics, and *artificial intelligence* (AI) ([1], [2]). This paper addresses a specific issue: the possibility of applying the AI methodology to the *error-correction* procedure in the communication module integrated in general IoT devices. One of the latest 5G wireless techniques is considered as a case study: *Polar coding*. The general background is described in this section and following.

An impressive estimate is that the modern IoT would consist of multiple billion objects in two or three years. A large portion of IoT involves *mission-critical objects* like communications, controllers, and actuators, which are the fundamental units in most IoT systems. For these applications, the signals for remote-control must be swift and reliable. The swiftness implies that the protocols of *automatic repeat request* (ARQ) are not appropriate. Therefore, the reliability needs to be attained with the aid of the *error correction coding* (a.k.a. *channel coding*) [3]. In modern coding theory, one of the state-of-the-art schemes is the *Polar codes* proposed by Arikan [4]. The present work focuses on small Polar codes for small IoT systems. Currently, the polar codes are mainly used for the 5G *new radio* (NR) *control channel*. This is particularly important for the remote-control procedure in most small IoT devices.

The diversity of small IoT devices would post a requirement for versatility of channel coding methods. Most decoding schemes are implemented in an iterative manner. However, compact and parallel schemes are desirable for the small devices with limited computing capacity and/or limited storage capacity. The present work describes the decoding approach for Polar codes by means of *machine learning* (ML) [5].

Nowadays, ML has been recognized as the core infrastructure of modern AI systems. It is highly interesting in investigating the feasibility of applying ML to deal with channel coding. In this paper, we report our practice and experience of applying the ML methodology to decode the non-systematic and systematic Polar codes.

The rest of this paper is organized as follows. In Section II, the system model and the Polar codes are reviewed. Then, in Section III, machine learning is introduced. Next, in Section IV, the decoding experiments with a neural network are reported and discussed. The conclusion is included in Section V.

Consistent with the notations in most papers in Polar codes, we use $G_P(n,k)$ to represent the generator matrix of the Polar code with codeword length n and user word length k. Accordingly, in total there are 2^k codewords.

II. THE SYSTEM MODEL AND POLAR CODING

The present work is concerned with digital wireless communication systems. The block diagram of this system is presented in Fig. 1. In a generic way, the communication procedures of a digital signal can be described as follows. In the transmitter side, to focus on the core concept, we omit the line coding module and the source coding module. With this simplification, we start looking at a random user word that enters the channel encoder. Then the channer encoder creates a codeword. Next, digital modulation is carried out. The modulated word is transmitted over a noisy channel (e.g., the AWGN channel). At the receiver side, the noisy signal is demodulated first. Then the detector makes a hard decision or soft decision. The former is based on the Hamming distance and the output is discrete, whereas the latter is based on the Euclidian metrics in the signal space. At this point, the output of the detector is a corrupted codeword. Next, this corrupted codeword enters the channel decoder. The channel decoder consists of two parts. In concept, the first part could be a brute-force algorithm that seeks the closest codeword from the corrupted codeword, keeping in mind that almost every practical code has its highly crafted algorithm. The second part is the algorithm that maps the found codeword to a user word.

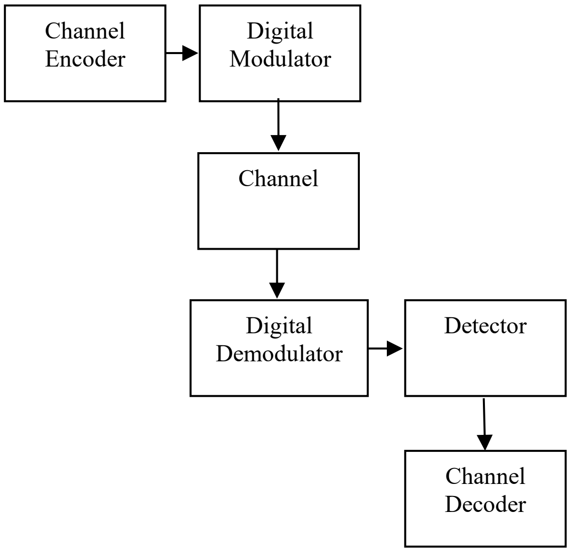

Figure 1. The digital wireless communication system.

In the discipline of error correction coding, the Polar code represents one of the most remarkable achievements. The Polar code is an elegant example of the linear block codes. Theoretically, it is the first provably capacity-achieving code, rather than demonstratively as other modern codes. In the literature, most reported Polar codes were *non-systematic*. For example, the following *non-standard* generator matrix was presented in [6],

$$G_P(8,5)=\begin{pmatrix}1&1&0&0&0&0&0&0\\1&1&1&1&0&0&0&0\\1&1&0&0&1&1&0&0\\1&0&1&0&1&0&1&0\\1&1&1&1&1&1&1&1\end{pmatrix}. \qquad (1)$$

Starting from the first order and second order of *Reed-Muller code* of length 16, we can also construct a non-standard generator for the Polar code of length 16:

$$G_P(16,8)=\begin{pmatrix}1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\1&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0\\1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0\\1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\end{pmatrix}. \qquad (2)$$

The Polar code is linear. According to coding theory, any non-systematic linear code can be converted to a *systematic* code, by converting the non-standard generator matrix to a *standard* form [7, Theorem 5.5]. With the procedure presented in [7, pp. 50-52], we can convert the above matrices to the standard forms, denoted as $G_{PS}(8,5)$ and $G_{PS}(16,8)$, respectively:

$$G_{PS}(8,5)=\begin{pmatrix}1&0&0&0&0&1&1&1\\0&1&0&0&0&1&1&1\\0&0&1&0&0&0&1&0\\0&0&0&1&0&1&0&0\\0&0&0&0&1&0&0&1\end{pmatrix}, \qquad (3)$$

$$G_{PS}(16,8)=[I_8 \mid B_8], \qquad (4)$$

where I_8 is the identity matrix of order 8, and

$$B_8=\begin{pmatrix}1&1&1&1&1&1&1&0\\1&1&0&0&0&0&0&1\\1&0&1&0&0&0&0&1\\1&0&0&1&0&0&0&1\\1&0&0&0&1&0&0&1\\1&0&0&0&0&1&0&1\\1&0&0&0&0&0&1&1\\0&1&1&1&1&1&1&1\end{pmatrix}. \qquad (5)$$

In Section IV, we will show that the ML decoder performs better for the systematic Polar code.

III. Machine learning and Neural Network

Currently there is an enthusiastic interest in machine learning (ML) or its advanced variant, deep learning (DL) [5]. Since the modern MD/DL discipline is built upon the *neural networks* (NN), the key insight can be gained by understanding the intrinsic principle of NN. There is a hierarchy of taxonomy. In general, NN can perform the *supervised training* or *unsupervised training*. NN can be applied to *pattern classification* or *function approximation*. For the present problem, we mainly address the aspects of NN for function approximation and supervised training. In this context, the NN methodology can be considered as a variant of mathematical regression. To simplify the concept, let us consider the linear regression. The NN discipline employs the following formula to approximate the unknown function in the form:

$$x^* = h(wy+c), \qquad (6)$$

where w and c are commonly called *weight* and *bias*, while $h(\bullet)$ is the (composite, chained) *activation* (transfer) function in the context of NN. The goal of the training process in NN is just to find the values of (w,c). After that, for any new values of y, the formula (6) can be directly used to calculate the corresponding target x^*.

In principle, two types of datasets are needed for the supervised training. In the context of the present work, one dataset consists of the original user words and the other dataset contains the corrupted codewords. These two datasets can be obtained through field measurement or a simulation. Note that our current interest is in decoding. Therefore, the corrupted codewords are used as the input dataset to NN, while the original user words are employed as the target dataset of NN. For a standard NN, the input data is typically formatted as an $R \times Q$ matrix, where R is the number of elements in each sample, while Q is the number of total samples. In the current problem, R is just the length of a single corrupted codeword. Q is dependent on the dimensions of NN. As a general rule to mitigating overfitting, the number of input items should be larger than the number of total unknown weight and bias in the concerned NN.

In this study, we designed a *feed-forward neural network* (FNN) [5, Ch. 6]. This FNN consists of three hidden layers, with 64, 32, and 16 neurons, respectively. Given the codeword length, the number of total unknowns can be derived accordingly. In the experiments, we used more samples than the number of total unknowns to avoid overfitting.

The standard NN consists of three processes: *training*, *validation*, and *testing*. First, in the training stage, the NN uses the available data to adjust the weights and bias. Secondly, in the validation stage, the NN tries to tune the *hyper-parameters* (HPs) up. One of the most common HPs is just the *weight decay*. Another example of HPs is the polynomial degree in the NN involving nonlinear regression. Thirdly, in the testing stage, a set of new data samples are used to verify the established mapping by NN. In our experiments, over the entire dataset, 70%, 15%, and 15% of samples were used for training, validation, and testing, respectively.

IV. Decoding Experiments

To assess the decoding functionality of the configured FNN, we conducted extensive experiments. Both FNN and the channel simulation are implemented in Matlab (R2018b).

In the experiments, the *Levenberg-Marquardt* (LM) algorithm was adopted as the training method. In the context of FNN, the LM algorithm is usually the first choice [5]. For activation functions, all three hidden layers use the *log-sigmoid* function and the output layer uses the pure linear function. The log-sigmoid function is defined as follows:

$$f(t) = \frac{1}{1+\exp(-t)}. \qquad (7)$$

The main operating parameters of this FNN are listed in Table I, where trainParam.epochs is the maximum number of iterations to train, trainParam.max_fail is maximum validation failures, and trainParam.min_grad is the minimum performance gradient.

TABLE I. Main Parameters in FNN

Parameters	Value
`trainParam.epochs`	200
`trainParam.max_fail`	10
`trainParam.goal`	10^{-6}
`trainParam.min_grad`	10^{-6}

A. *The Non-Systematic Polar (8, 5) Code*

This code is generated from the matrix presented in (1).

A.1 Training

The performance of *mean square error* (MSE) of training, validation, and testing are illustrated in Fig. 2. Next, the corresponding error profile is presented in Fig. 3, where the ordinate represents the number of errors that falls within each interval on the abscissa. It is observed that most instances fall in a very narrow error interval of 5×10^{-5}.

According to one of the main criteria of NN, the training tends to be perfect if the value of index R is close to 1. Basically, the index R characterizes the match between the provided values and the predicated values. In the context of decoding an error correction code, the provided values are the given corrupted codewords, while the predicated values are the use words regenerated by the FNN. As shown in Fig. 4, the index R is almost perfect. Accordingly, it is concluded that the communication system has been successfully trained by this FNN.

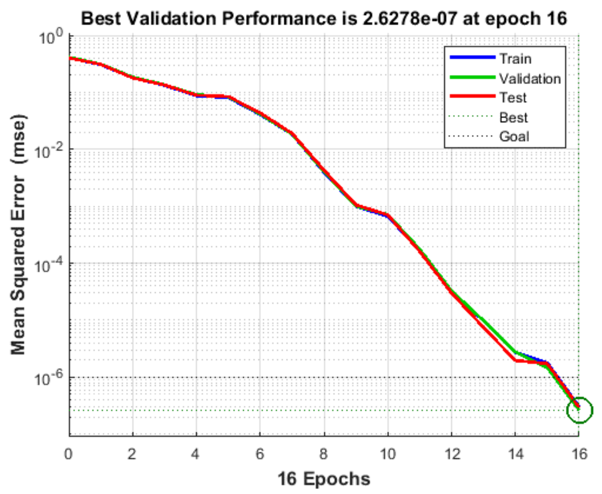

Figure 2. Mean square error (MSE) performance.

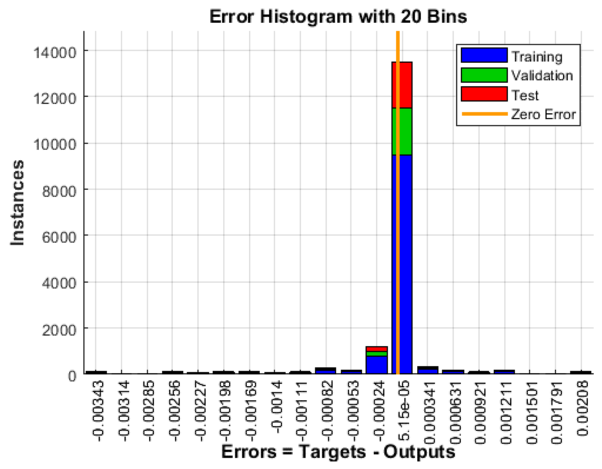

Figure 3. Histogram of errors.

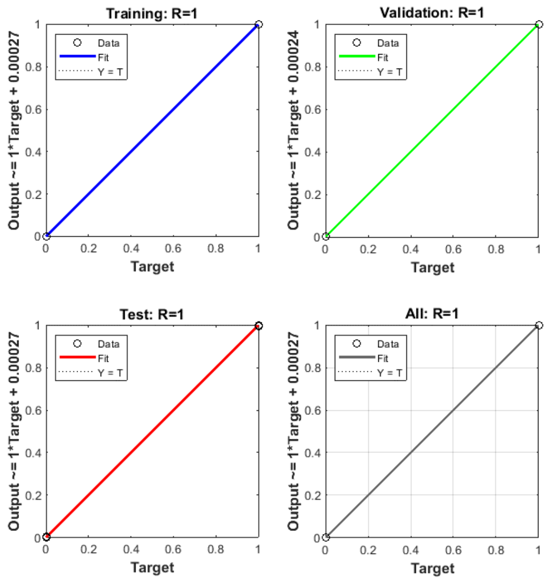

Figure 4. Scatter plots.

A.2 Decoding Test

In the ML procedure, the test stage follows the training stage. As one of the most important axioms of ML, the datasets used for testing should have never been used in the training stage. For this purpose, we run simulations to generate different random data pairs of user words and corrupted codewords were independently generated for use. The SNR is used as one of the main parameters in simulations. For each SNR, three datasets are tested. Each dataset contains 3369 samples. The performance is evaluated with the *bit error rate* (BER), which characterizes the difference between the estimated user words and the original user words. The profile is illustrated in Fig. 5.

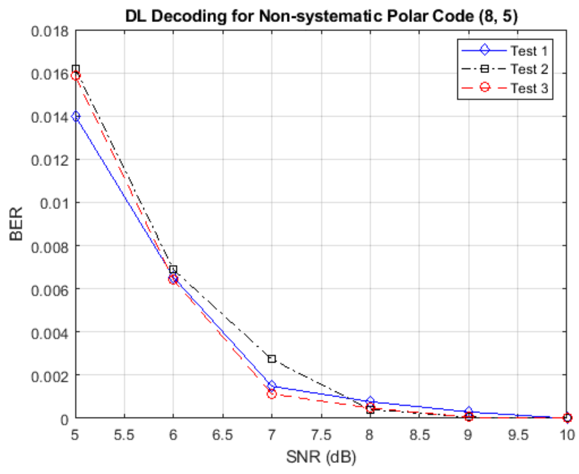

Figure 5. Performance of ML decoding.

B. *The Systematic Polar (8, 5) Code*

This code is generated from the matrix presented in (3).

B.1 Training

In this experiment, all parameters are the same as those used for the non-systematic case. The performance of MSE of training, validation, and testing are illustrated in Fig. 6. We note that the number of epochs is reduced from 16 to 9, a significant increase in training efficiency. Furthermore, as shown in Fig. 7, the index R is almost perfect. Accordingly, this code has been successfully trained by the FNN.

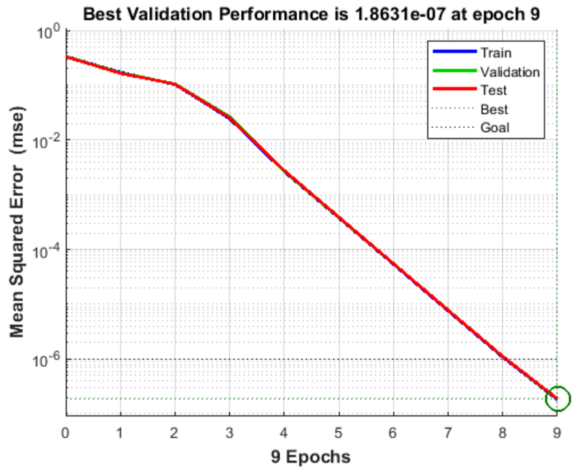

Figure 6. Mean square error (MSE) performance.

B.2 Decoding Test

Similar to the non-sysemstic code, for each SNR, three datasets are tested. Each dataset contains 3369 samples. The profile of BER against SNR is illustrated in Fig. 8. Compared with Fig. 5, it is observed that the BER is significantly reduced in the low SNR regime. For example, at SNR = 5 dB, the BER in Fig. 8 is around 6×10^{-3}, while in Fig. 5 the corresponding BER is about 1.5×10^{-2}.

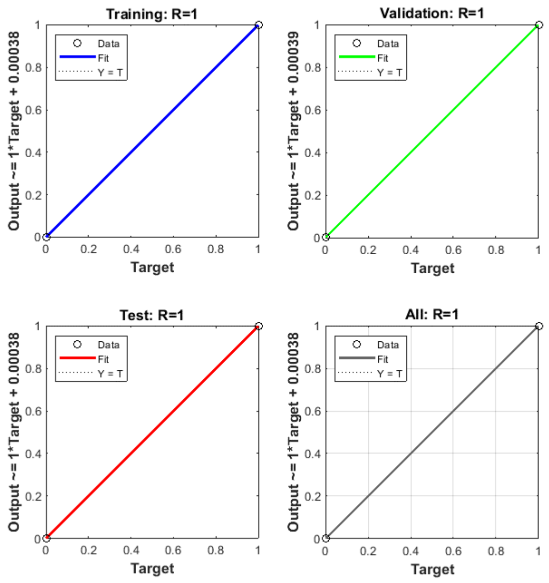

Figure 7. Scatter plots.

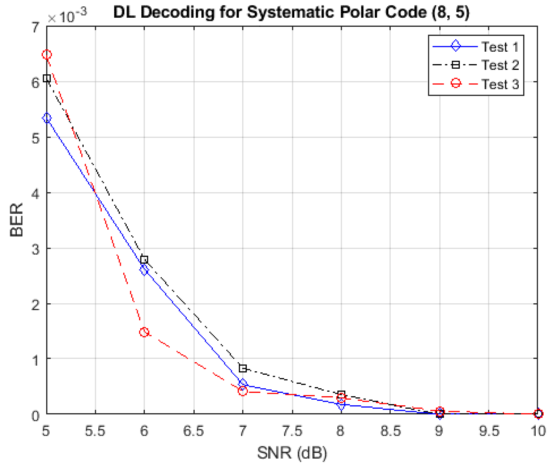

Figure 8. Performance of ML decoding.

C. *The Non-Systematic Polar (16, 8) Code*

This code is generated from the matrix presented in (2).

C.1 Training

In this experiment, all parameters are the same as those used for the Polar code (8, 5). The performance of MSE of training, validation, and testing are illustrated in Fig. 9. The training is stopped at the specified threshold with 17 epochs. As shown in Fig. 10, the training is successful.

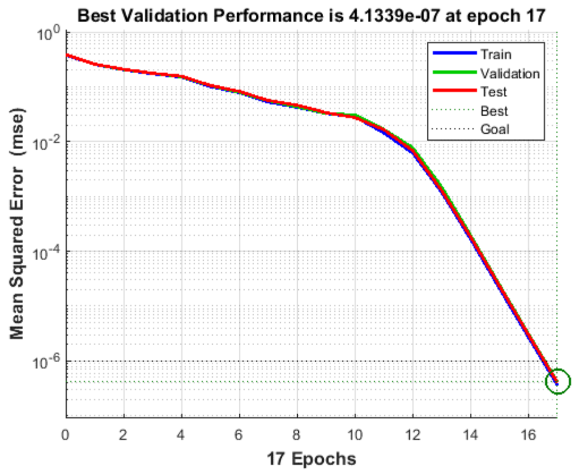

Figure 9. Mean square error (MSE) performance.

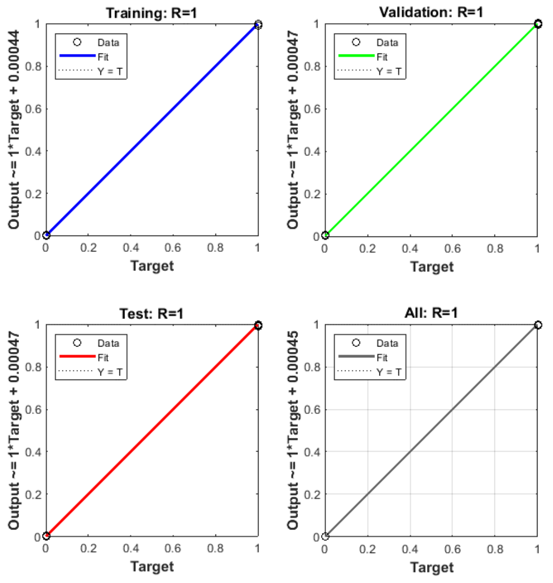

Figure 10. Scatter plots.

C.2 Decoding Test

The profile of BER vs. SNR is presented in Fig. 11. For each SNR, three datasets are tested. Each dataset contains 3369 samples. It is similar to the profile of non-systematic Polar code (8, 5) as shown in Fig. 5.

D. *The Systematic Polar (16, 8) Code*

This code is generated from the matrix presented in (4).

D.1 Training

In this experiment, all parameters are the same as those used for the non-systematic case. The performance of MSE of training, validation, and testing are illustrated in Fig. 12. The training is stopped at the specified threshold with 12 epochs. As shown in Fig. 13, the training is successful.

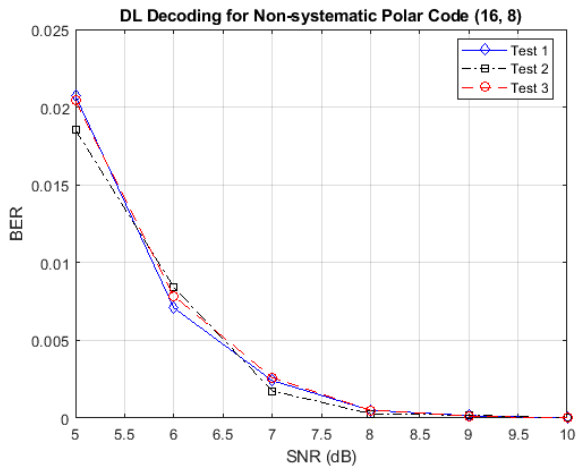

Figure 11. Performance of ML decoding.

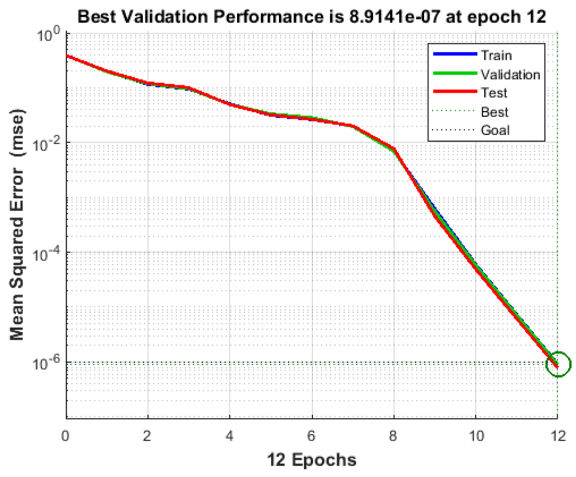

Figure 12. Mean square error (MSE) performance.

D.2 Decoding Test

Similar to the non-sysemstic code, for each SNR, three datasets are tested. Each dataset contains 3369 samples. The profile of BER against SNR is illustrated in Fig. 14. Compared with Fig. 11, it is observed that the BER is significantly reduced in the low SNR regime. For example, at SNR = 6.5 dB, the BER in Fig. 14 is around 2×10^{-3}, while in Fig. 11 the corresponding BER is about 5×10^{-3}.

V. CONCLUSION

This paper investigates the feasibility of applying the ML methodology to the error-correction procedure in the remote-control module integrated in compact IoT devices. In this study, we use the neural network (NN) to decode four small Polar codes: non-systematic (8, 5) and systematic (8, 5), as well as non-systematic (16, 8) and systematic (16, 8). It is shown that, with an appropriately configured NN, the decoding can be well done. So those long iterative decoding approaches could be avoided in small IoT devices.

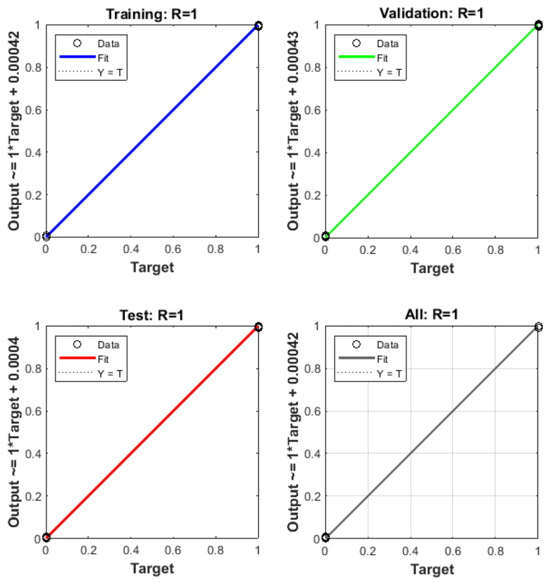

Figure 13. Scatter plots.

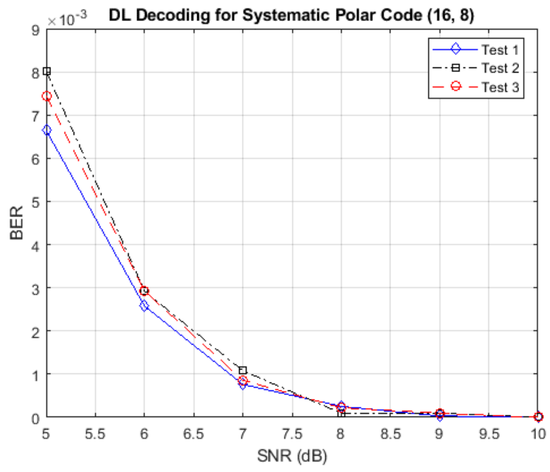

Figure 14. Performance of ML decoding.

REFERENCES

[1] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Enge, and L. Ladid, "Internet of Things in the 5g era: enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510-527, 2016.
[2] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on 5G networks for the Internet of Things: communication technologies and challenges," *IEEE Access*, vol. 6, pp. 3619-3647, 2018.
[3] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge University Press, 2003.
[4] E. Arıkan, "Channel polarization: a method for constructing capacity achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, pp. 3051–3073, July 2009.
[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
[6] E. Arıkan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 447-449, June 2008.
[7] R. Hill, *A First Course in Coding Theory*, Oxford University Press, 1986.