Deep Learning in Encoding and Decoding of Polar Codes

Chao Yuan¹,a, Chang Wu¹,b,*, Dan Cheng¹,c and Yang Yang¹,d

¹University of Electronic Science and Technology of China, Chengdu, China
²13076091292@163.com, ³changwu@uestc.edu.cn, ⁴danchenguestc@163.com, ⁵2654016764@qq.com

Abstract. In this paper, we apply neural networks to the encoding and decoding of polar codes. Polar codes in 5G communication which meets the simple and ultra-low latency can also be improved. This paper aims to improve the polar codes to achieve simplicity and hardware friendliness. After trained by the neural network, we get a simple math formula to estimate the reliability of the encoding in the polar codes, which can achieve the same effect as GA (Gaussian Algorithm). In terms of decoding, by using NND (Neural Network Decoding) to decode polar codes, we achieve almost the same effect as the conventional SC (Successive Cancellation) algorithm. In order to achieve further improvements, we use the neural network to optimize BP (Belief Propagation) algorithm. Thanks to it, a small number of iterations can be enough to decode the polar codes, even with an iteration of one. The simulation shows that though the early stage of training is full of difficulties, the result can be used to achieve one-step and fool-type decoding.

1. Introduction
In 1943, W.Mcculloch and W.Pitts came up with an NN (neural network) idea to sequentially solve the problem by simulating the human brain architecture [1]. In 1990s, a variety ideas around the use of NN for decoding emerged. The concept of deep learning stems from the study of artificial neural networks. The advantage of deep learning is the use of unsupervised or semi-supervised feature learning and hierarchical feature extraction to efficiently replace manual feature capture. Deep learning is a new field in machine learning research, and it aims to establish a neural network that simulates analytic learning of human brain.

In 2008, Erdal Arikan first proposed the concept of channel polarization at the International Symposium on Information Theory (ISIT) conference [2]. Polar codes have a deterministic construction method and are the only known channel coding methods that can be rigorously demonstrated to "reach" the capacity of the channel.

However, the computational complexity over the code reliability estimation and decoding algorithms of polar codes is very huge, which requires strong mathematical foundation. At the same time, there exists a problem of big calculation which is linear with the block length. Thus, systems with different parameters can not be accepted, such as a system with different block length and code rate, and they are not even feasible for low latency, real-time encoder or decoder. As 5G control channel coding standard, Polar codes are required to gain low latency, but the traditional algorithm is not perfect to meet the requirement, so we have to improve the algorithm. In recent years, boosted by the great achievement people made on deep learning and parallel computing ability of computer, a
series of problems the polar codes are facing can be solved by using a powerful deep learning network. This is the trend of development, and it is also the necessity of development.

This paper mainly aims to use deep learning networks to solve some of the problems of polar codes. The results shown below are based on my attempts, and more further work still needs more scholars to study together. In this paper, the polarized degree of polar codes, which is also the reliability of the polar codes, is calculated before decoding. The traditional way to calculate the reliability is DE (density evolution) and GA algorithms [3], these algorithms are complex and have large latency. By using the deep learning to train the polar codes, we can use curve fitting to conclude a simple formula to estimate the reliability, which can also be used to deal with the complex attached parts created by polarizing codes. Then, when we decode the polar codes, we can use deep learning to achieve fool-type, one-step decoding. If we have decoding training matrix in advance, once we receive the data, we only have to do simple multiplication and addition, but we can achieve the same effect with traditional SC decoding. In order to achieve better decoding results, we use the deep learning to modify the parameters in BP decoding algorithm, which really makes a difference.

The rest of this paper is organized as follows: The section II introduces the basic knowledge of deep learning and polar codes and the principles to be mastered. The section III shows the application of NN learning in decoding reliability estimation and simulation results. The section IV presents the application of NN when decoding the polar codes. The section V is the summary of this paper.

2. Introduction of polar codes and deep learning

For deep learning, the idea is to stack multiple layers, that is, the output of each layer is the input of next layer. In this way, the input information can be expressed hierarchically.

When decode the polar codes, we must first distinguish the reliability of $N$ splitted channels, that is to distinguish which channel is reliable and which channel is unreliable [4]. There are three commonly used methods for measuring the reliability of each polarized channel: the Bhattacharyya Parameter method, the DE method, and the GA method.

Arikan gave a decoding algorithm of polar codes in his paper, that is, successive cancellation (SL) decoding algorithm [5]. For polar codes, when the code length tends to infinity, the channel polarization is more complete. SC decoding algorithm is a greedy algorithm, at each layer of the tree, we only search for the optimal path then we go to the next level, so the error can not be modified. SCL decoding algorithm is to improve the SC algorithm [6].

In addition, there are spherical decoding algorithms and BP decoding algorithms [7]. In the proposed BP decoder, The reliability of the transmission messages can be enhanced by increasing the checkout node. In particular, it is possible to correct the error messages transmitted from the frozen node. The coding of the polar codes can be divided into $n$ stages, and at each stage, the size of processing sets (PS) is $N/2$. Each PS has two input and output nodes[8].

3. Deep learning and polarization endoding

1): Build a deep learning environment. Keras is a API of high-level neural network. It is written by pure Python language and based on Tensorflow, Theano and CNTK backend. As for the specific building environment, you can refer to the machine learning web: http://keras-cn.readthedocs.io/en/latest/.

2): Calculate Bhattachayya parameter of input data. Then the transition probability of the channel $w^N: x^N \rightarrow y^N$ is:

$$w^N(y^N | x^N) = \prod_{i=1}^{N} w(y|x)$$

Bhattacharyya parameter can be expressed as :

$$Z(w) = \sum_{y \in \mathcal{Y}} \sqrt{w(y|x)w(y|x)}$$

(1)
Then, we need to calculate the Bhattacharyya parameter of several sets of input data. For example, the number of input bits \( N \) is: 16, 32, 64, 128… \( 2^n \) ( \( n=\log_2 N \) ). The Bhattacharyya parameter corresponding to input data \( x_i^N \) is sorted as ascending order and recorded as \( z_i^N \).

3): Convert the ordered subscript from 1 to \( N \) of input \( x_i^N \) to binary number \( b_i \) \((i=[1,N])\). Then build a model of deep learning and enter the training data: \( X_{\text{train}}=b_i, Y_{\text{train}}=z_i^N \).

The number of bits of input data is \( N \). Generally speaking, we use the training networks with three layers(2\( N \)-4\( N \)-2\( N \)). You can also continue to correct the training network during the training process.

The NND flow chart as shown in Figure 1:

![Figure 1. The NND flow chart about the NN of polar codes](image)

By constantly adjusting the parameters corresponding to the deep learning model and training the data, and using the principle of \( \beta \)-expansions [9], a simple and efficient formula for estimating the reliability of polar code can be obtained:

\[
k_i = \sum_{i=0}^{N} b_i \varphi^i + c
\]

(3)

where \( k_i \) is the estimated value of reliability, \( \varphi \) and \( c \) are constants and their values are not the same in different channel environments. In the Gaussian white noise channel environment, \( \varphi = 1.1777, c = 1.1133 \)

4): How to use this formula.

For example, the code length of polar codes is \( 2^5 = 32 \), and the binary expression whose subscript is 5 is 00101. A calculation of one of them is shown in formula (14). When all the estimated values are calculated, the value of the corresponding index position is sorted by descending order.

\[
K_s = \sum_{i=1}^{5} b_i (\sqrt{\varphi})^i + c = 1*\left(\sqrt{\varphi}\right)^5 + 0*\left(\sqrt{\varphi}\right)^4 + 1*\left(\sqrt{\varphi}\right)^3 + 0*\left(\sqrt{\varphi}\right)^2 + 0*\left(\sqrt{\varphi}\right) + c
\]

(4)

According to the simulation and BPE (Bhattacharyya Parameter Estimate), the comparisons are as follows Figure 2:

![Figure 2. The simulation of BPE and Deep learning in different frames](image)
The simulation data are shown as follows: There are two group data and their frames length are \(N=512\) and \(N=1024\) respectively. The total number of frames is \(M\). The signal through the Gaussian white noise channel. The modulation method is BPSK. And the encoding efficiency of input is 0.5, in other words, the effective length of polar codes is 0.5\(N\). We adopt successive cancellation list (SCL) to decode, where \(L=8\). And we add the cyclic redundancy check (CRC) at the end, the length of CRC is 17. The signal-to-noise ratio (SNR) required for reliability of the BP method and deep learning method is compared under the circumstance of which the frame error rate is 0.001.

The simulation results show that: the DL (deep learning) method obtained by machine learning training and the BPE method need similar SNR under the same conditions, and the BPE is a little better than DL. But the calculation of DL is simple and easy to understand, and its calculation amount is much smaller than BPE. When the code length and bit rate are relatively large, the DL method can realize the instant implementation of encoder/decoder for low latency. But this method is not of universality. It needs to do machine learning in advance according to different channel environment and communication scenarios, and then correct the parameters involved in this method. Only when we get better parameters, this method can be comparable to the traditional algorithms.

4. Deep learning and polar code decoding

In this part, we want to use NN to decode noisy code-words. At the transmitter, k-bits information is encoded into code-words of length \(N\). The encoded bits are modulated and transferred on the noisy channel. At the receiver, we can receive the code-words with noise. And the mission of decoder is to recover the corresponding information bits. Compared with iterative decoding, NN finds its estimated value through only one layer. It is a arduous and expensive task for deep learning to get good trained data. But using NN for channel encoding is special, because we deal with artificial signals. Therefore, we can generate as many training samples as possible. Moreover, the desired NN output, also denoted as label, is obtained for free because if noisy code-words are generated, the transmitted information bits are obviously known [10]. For simplicity, binary phase-shift keying (BPSK) modulation and additive white Gaussian noise channel (AWGN) are used. The other information of channels can be used directly, and the flexibility may be a particular advantage of NN-based decoding.

The examples of loss function are mean square error (MSE) and binary cross entropy (BCE):

\[
L_{MSE} = \frac{1}{K} \sum_{i} (b_i - \hat{b}_i)^2
\]

\[
L_{BCE} = -\frac{1}{K} \sum_{i} b_i \ln(\hat{b}_i) + (1 - b_i) \ln(1 - \hat{b}_i)
\]

Where \(b_i\) is the input bit that is the encoded bit of polar codes, \(\hat{b}_i \in [0,1]\), \(\hat{b}_i\) is the decoded bit estimated by NN, \(\hat{b}_i \in [0,1]\).

This processing step can also be implemented as an additional layer without requiring any training parameters. Note that in this case, the noise variance must be known and provided as an additional input of NN. We use Keras3 as the advanced abstract front-end of keras3. It allows rapid deployment of NN from the very abstract perspective of the Python programming language, thus hiding much of the underlying complexity [11].

We train our NN decoder at so-called “epochs”. At each epoch, the gradient of the loss function is calculated over the entire training set \(X\) by using Adam, a stochastic gradient descent optimization method [12]. Because the noise floor in our architecture will produce a new noise in each use, the NN decoder will never see the same input twice. For this reason, although the training set has a finite size of \(2k\) code-words, we can train the data on a basic infinite training set by simply increasing the max training number of epoch (Mep). However, this makes it impossible to distinguish whether the the neural network has been improved through a large number of training samples or more optimization iterations.

The simulation result is shown in Figure 3. From the results, NN decoder is little better than the SC, but we need huge number of Mep which means we must spend a lot of time to train the NN. If the length of word-codes is long, we have to face the curse of dimension. So, this way of NN only fit to
short word-codes. There is still a gap between the effect of direct decoding by traditional neural network and the decoding of polar codes. To achieve a better training effect, we study the combination of neural network and BP algorithm.

![Graph of the BER of different Map MND](image)

**Figure 3.** The simulation of deep learning on decode of polar codes with 64 bit-length codes with code rate $r = 0.5$ in AWGN channel.

For the BP algorithm, in order to reduce the computation and make it easy to realize in hardware, we compare the LDPC decoding min-sum product algorithm (SPA). And we use the sum-min BP to train the neural network, because it is easy to realize.

The specific formulas are shown as follows (7-10):

$$L_{i,j} = \alpha_i \cdot \text{sign}(L_{i,j+1}^{-1}) \cdot \text{sign}(L_{i+2,j+1}^{-1} + R_{i+2,j}^{-1}) \cdot \min(|L_{i,j+1}^{-1}|, |L_{i+2,j+1}^{-1} + R_{i+2,j}^{-1}|) + \alpha_i'$$

$$L_{i+2,j+1} = L_{i+2,j+1}^{-1} + \beta_i \cdot \text{sign}(L_{i,j+1}^{-1}) \cdot \text{sign}(R_{i,j}^{-1}) \cdot \min(|L_{i,j+1}^{-1}|, |R_{i,j}^{-1}|) + b_i'$$

$$R_{i,j} = \chi_i \cdot \text{sign}(R_{i,j}^{-1}) \cdot \text{sign}(L_{i,j+1}^{-1} + R_{i,j}^{-1}) \cdot \min(|R_{i,j}^{-1}|, |L_{i,j+1}^{-1} + R_{i,j}^{-1}|) + \epsilon_i'$$

$$R_{i+2,j} = R_{i+2,j}^{-1} + \delta_i \cdot \text{sign}(L_{i,j+1}^{-1}) \cdot \text{sign}(R_{i,j}^{-1}) \cdot \min(|L_{i,j+1}^{-1}|, |R_{i,j}^{-1}|) + d_i'$$

where $\alpha, \beta, \chi, \delta, a, b, c, d$ are all the coefficients of each belief propagation.

Here, $L_{i,j}, R_{i,j}$ represent the messages at t-th iteration from the left-to-right and right-to-left based on log-likelihood ratio respectively. According to the above formulas, the neighbouring nodes can propagate their associated messages iteratively, and these messages can be updated by PS. After the BP decoder reaches the preset maximum iteration, we can determinate the decoding bits and output by the hard decision based on the overall LLR of the node in formula (11):

$$\hat{u_i} = \text{sign}(\text{LLR}_{i,j}^{\max,\text{iter}}) = \text{sign}(L_{i,j}^{\max,\text{iter}} + R_{i,j}^{\max,\text{iter}})$$

The purpose of this paper is to use the neural network to correct the numerical value of each iteration of $\alpha, \beta, \chi, \delta, a, b, c, d$, thus, by making the BP algorithm in a specific channel environment achieve a better effect.

The specific calculation process is as follows:

1. Calculate the maximum log-likelihood ratio $\text{LLR}(c)$ of the signal received through channel.

2. Prepare some parameters. The frozen bits in the reliability result calculated by the neural network, are recorded as $frz = \{frz_1, frz_2, \cdots, frz_{n_{frz}}\}$. $\alpha, \beta, \chi, \delta$ are all initialized to 1, $a, b, c, d$ are all initialized to 0. And we define the maximum number of iterations as $T_{\text{iter}}$.

3. Start BP iteration algorithm:

$$L_{i,j}^0 = 0, R_{i,j}^0 = 0$$

(12)
\[
\begin{align*}
\text{if} : (j = 1) \& (i \in frz) \rightarrow R_{i,j}^t &= \infty \\
\text{else : when}(j = 1 + m) \rightarrow L_{i,m+1}^t &= LLR(t) \\
\text{if}(L_{i,m+1}^t + R_{i,m+1}^t \geq 0) \rightarrow \hat{x}_i = 0, \text{else} \rightarrow \hat{x}_i = 1 \\
\text{if}(L_{i,j}^t + R_{i,j}^t \geq 0) \rightarrow \hat{u} = 0, \text{else} \rightarrow \hat{u} = 1
\end{align*}
\]  

(13)

(14)

When the number of iterations is smaller than the maximum, we update the values of \( L_{i,j}, R_{i,j}, \hat{u}, \hat{x} \) constantly.

It is time to stop iteration while the matrix meets the conditions: \( \hat{u}G = \hat{x} \), where \( G \) is the generated matrix of polar code.

4. When it meets the conditions: \( \hat{u}G = \hat{x} \), we record the values of \( L_{i,j}, R_{i,j}, t_s \) is the iteration value at this time.

We use encoded \( u \) to derive the corresponding results: \( u \rightarrow R_{i,j}, L_{i,j}, \langle L_{i,j}, R_{i,j} \rangle \) can correct the value of \( \alpha, \beta, \chi, \delta, a, b, c, d \), through the neural network training, where \( t = t_s - 1 \). Keep training, when \( t = 0 \), stop training.

Record the corrected value of \( \alpha, \beta, \chi, \delta, a, b, c, d \) and decode. When \( \hat{u}G = \hat{x} \), the decoding is finished. At this time, the whole decoding process has been finished.

The complete algorithm flow chart is as follows in Figure 4.

\[ \begin{array}{c}
X_i^N \\
G \\
B \\
P \\
N \\
BPNN
\end{array} \quad u = xG + n_d \]

**Figure 4.** The complete algorithm flow chart of BPNN.

In this chart, \( G \) is generator matrix. \( \delta_n \) is the channel condition. \( F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \) represents the Kronecker product [13].

The result of simulations is depicted in Figure 5:

\[ \begin{array}{c}
\text{the BER of different decode of polar code: SC, BP, BPNN} \\
\text{the BER of different decode of polar code: SC, BP, BPNN} \\
\end{array} \]

**Figure 5.** The simulations of different decoding with SC, BP, BPNN in 1024 bits of a frame (10000 frames) and \( r \) is 0.5, the modulation mode is BPSK, and the channel is AWGN.
By comparing the BP decoding with the BPNN (BP+NN) which is the BP through the neural network, we find that BPNN is better than BP especially at the larger SNR. The BPNN training work can be done well in advance, so the decoding of polar codes in BPNN decoding only needs a few times of iteration, even one iteration can realize decoding. It can reduce a lot of computation and lower time delay, and make the hardware implementation be easy at the meantime. BPNN also makes the encoding and decoding of polar codes be realized in one step, so that the polar codes are easy to implement in hardware and has ultra-small time-delay. So it is very suitable for 5G scene requirements.

5. Conclusion

Through the study of the neural network effects on polar codes’ encoding and decoding, this paper has the following results. At the time of encoding, we obtain a simple and efficient estimation method with low latency by training the reliability of polar code. This method has no complicated calculation and is easy to understand and comparable to GE. When decoding, through the direct neural network decoding we can decode in one step. And this decoding has the same effect with SC decoding. When the requirement is not high or the channel environment is better, we can use the NN decoding. If the better bit error rate is needed, we can use BP+NN to achieve the BP decoding with one iteration by correcting the BP parameters by constant training. In conclusion, BPNN is a very suitable method. And the polar codes’ encoding and decoding will be simple and practical after neural network training. It can implement by only one fool-style operation, and it can realize the high efficiency and low latency in hardware. This paper is based on special environment, and different environment needs neural network training. And with the length of code-words becomes larger and larger, the neural network becomes more and more difficult and it needs continuous study.

References

[1] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115-133, 1943.
[2] E. Arikan, “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, Jul. 2009.
[3] I. Tal and A. Vardy, “How to construct polar codes,” IEEE Trans. Inf. Theory, vol. 59, no. 10, pp. 6562-6582, 2013.
[4] N. Hussami, S. Korada, and R. Urbanke, “Performance of polar codes for channel and source coding,” in Proc. IEEE ISIT, 2009, pp. 1488-1492.
[5] I. Tal and A. Vardy, “List decoding of polar codes,” in Proc. IEEE Int.Symp. Inf. Theory, 2011, pp. 1-5.
[6] P. Trifonov, “Efficient Design and Decoding of Polar Codes,” IEEE Transactions on Communications, vol. 60, no. 11, pp. 3221-3227, November 2012
[7] Vikalo, H., and B. Hassibi. "On the sphere-decoding algorithm II. Generalizations, second-order statistics, and applications to communications." IEEE Transactions on Signal Processing 53.8(2005):2819-2834.
[8] B. Yuan and K. K. Parhi, “Architecture optimizations for BP polar decoders,” in Proc. 38th IEEE ICASSP, 2013, pp. 2654-2658.
[9] Parry, W., “On the β-expansions of real numbers”, Acta Mathematica Academiae Scientiarum Hungaricae, 11: 401-416, 1960.
[10] Gruber T, Cammerer S, Hoydis J, et al. On Deep Learning-Based Channel Decoding[J]. 2017.
[11] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org
[12] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” CoRR, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980
[13] N. Hussami, S. B. Korada, and R. Urbanke, “Performance of polar codes for channel and source coding,” in Proc. 2009 IEEE Int. Symp. Inform. Theory, (Seul, Korea), p. 1488-1492, 2009.