International Journal of Advanced Research and Review

www.ijarr.org

CONVOLUTIONAL NEURAL NETWORK-BASED CHANNEL ENCODER CLASSIFICATION

[1]Shaik Dilshad, [2]G.Lakshmikanth,

[1]PG Scholar, Dept.of.CSE, Sree Rama Engineering college,Karakambadi road, Tirupati-517507.

[2]Assistant Professor, Dept.of.CSE, Sree Rama Engineering college,Karakambadi road, Tirupati-517507.

Abstract

Channel encoders are essential in digital communication systems for correcting channel-induced random errors. In most cases, the receiver has access to details on the transmitting end's channel encoders, including their kind and characteristics. The kinds and characteristics of encoders may only be known to a limited extent or not at all in non-cooperative situations, such as military communication systems. In this research, we investigate the possibility of using a deep learning strategy to categorize four distinct kinds of encoders: polar, block, convolutional, and Bose Chaudhuri-Hocquenghem (BCH). Our suggested method achieves classification accuracy surpassing 95% up to a bit-error-rate (BER) value of 0.03 using a convolutional neural network (CNN) model. Also, when the input sample length increases, the accuracy improves, according to the findings.

Index Terms—Convolutional neural network(CNN), channel encoder classification, deep learning, non-cooperative scenarios

INTRODUCTION

Forward error-correcting codes (FEC codes) are essential in digital communication for reducing the impact of random transmitter mistakes [1]. Decoding relies on having a firm grasp of the receiving end FEC encoders. Blind estimate of the channel encoder is necessary in certain cases, especially when the receiver does not have any previous information about the transmitter's encoder, which prevents proper decoding [2]. Afterwards, a number of novel algorithms and methods for blindly reconstructing channel encoders have been proposed. These innovations, meanwhile, bring with them new difficulties. In non-cooperative communication contexts, where decoding messages from unknown sources relies on precise reconstruction of the channel encoder, resolving these issues has substantial ramifications. Spectral efficiency may be improved by reducing channel resource consumption by blind parameter identification in channel encoders [1, 2]. Works like [2] and [3] have investigated the blind recovery of convolutional encoders using algebraic and dual-code characteristics. In [4] and [5], researchers examined blindly how to reassemble a pair of recursive systematic convolutional (RSC) encoders by using iterative expectation-maximization (EM) and the least-square approach.

While [7] used hamming weight distribution to identify encoder parameters, [6] relied on rank deficit of the data matrix. By capitalizing on the number of non-zero columns and non-zero elements in the column echelon form of the data matrix, the authors presented parameter estimation techniques for Reed-Solomon (RS) codes. A recent body of research investigated blind convolutional code recognition using convolutional neural networks (CNNs) with an accuracy level more than 90%. Current approaches are computationally intensive and vulnerable to low signal-to-noise ratios; examples are mathematical algorithms and rank-based methods. We find that our CNN model struggles with channel-encoded data that has sequential dependencies, doesn't have memory for long-range dependencies, and

may not even grasp the physics of channel encoding. A. What Came First and What I Did Importantly, current studies have mostly dealt with applying deep learning methods to identify specific FEC codes; however, there is a lack of work on classifying channel encoders using CNN. Utilizing deep learning techniques, this work aims to classify four channel encoders from an incoming noisy signal and evaluate the classification accuracy, or the probability of correct identification, under different bit error rate (BER) conditions. For this task, we take into account polar, Bose-Chaudhuri-Hocquenghem (BCH), convolutional, and Hamming encoders.

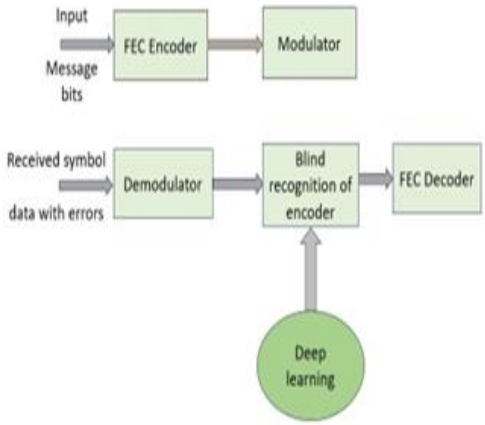

Fig. 1. Encoder classification process

CNN-BASED BLIND ENCODER CLASSIFICATION

Figure 1 shows the FEC encoder classification process. First, the FEC encoder is fed a string of bits denoted as b = [b1,b2,...,bk] that are created at random. A series of encoded data bits with a block size of n, indicated as c = [c1,c2,...,cn], is produced by this encoder, where each ci is an element of the Galois Field GF (2). The continuous information bits have a block size of k. Where k is the code dimension and n is the codeword length, the code rate is given by r = k/n. Modulation is applied to the decoded digital sequence prior to transmission via the communication channel. Binary phase-shift keying (BPSK) is one of the modulation methods that was chosen for the transmitter. The data bits, y = [y1,y2,...,yn], are extracted from the incoming signal by demodulation once it reaches the receiving end. In the next step, the data bits are sent to the FEC decoder to be recovered from their original state. Autonomous encoder identification or categorization using a convolutional neural network (CNN) model is the main emphasis of this study. Satellite communications, wireless fidelity (Wi-Fi), and mobile communication standards such as global system for mobile communications (GSM), code di vision multiple access (CDMA), and 5G new-radio (NR) all make heavy use of the channel encoders under consideration.

Dataset Generation

In our experimental design, four distinct coding systems are tested by generating datasets in MATLAB. We presuppose flawless frame synchronization and effective information signal demodulation at the receiving end. Therefore, we will pretend that the channel is AWGN (additive white Gaussian noise). Four FEC codes, one of which is polar code, are shown in the article. Channel polarization is the foundation of polar codes, which are the first codes to provide explicit evidence of achieving channel capacity for symmetric binary-input discrete memoryless channels (B-DMC). Encoding and decoding rely heavily on the dependability sequence, which is described by a generator matrix that is constructed recursively by the Kronecker product [9]. Block codes, on the other hand, work by dividing data into blocks of fixed size and processing each one separately. A Hamming block coding, often used for single-error correction, is selected for our experiment. Furthermore, BCH codes, which are also block codes, are well-known for their strong error correcting capabilities. They can effectively handle random

and burst mistakes in many applications. However, since it processes data in real time, a convolutional encoder differs from block codes. The encoded output is generated via bit-by-bit processing of input data in a shift register, which is then combined using modulo-2 additions. Commonly, block codes are represented as (n,k), where r = n−k is the length of the bits used for parity or redundancy. N0 is the length of the coding constraint, and (n,k,N0) is the notation used to express the convolutional code in this study. N is the codeword length, which is always an exponent of 2 (i.e. 2n (n ≥ 2)), and K is the symbol for Fig. 2. These polar codes are defined in this study and are designated as (N,K). Constraints on convolutional neural networks amount of bits used in the message. The quantity of frozen bits with dependability sequence Q is represented by the difference N−K [9].

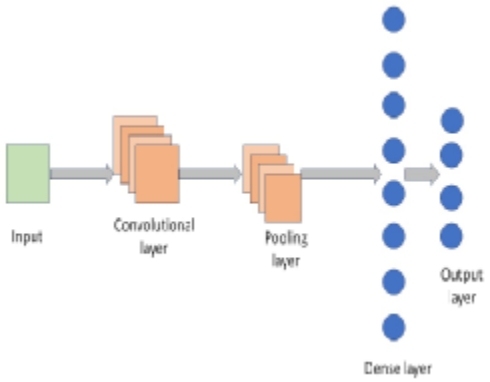

Fig. 2. Basic structure of CNN the number of message bits.

The difference N−K corresponds to the number of frozen bits with reliability sequence Q [9].

PROPOSED CNN MODEL

Section III. Convolutional Neural Network Model The main parts of a convolutional neural network (CNN) design are the input and output layers, as well as the convolutional, pooling, and fully-connected layers, as shown in Figure 2. Data in the form of images or grids may be fed into the input layer. Central to convolutional neural networks (CNNs) are the convolutional layers, which use a series of filters to extract information [11]. These filters detect patterns like edges, textures, and basic forms by performing convolution operations over the input data. Convolutional layers use filters, often called kernels, which are tiny matrices or tensors that convolve over the input data. We minimize the spatial dimensions of the feature maps in the succeeding pooling layers while keeping critical information. Methods like max-pooling and average-pooling, which are often employed in pooling, accomplish this decrease. Lastly, characteristics obtained from earlier layers are processed to a level where fully linked networks cannot keep up.

A softmax-based probability distribution is used to achieve this. Section A. Convolutional Neural Networks Forward propagation, weight up dates, loss computation, backpropagation, and initializing weights are all steps in training a convolutional neural network (CNN). The first step is to gather and prepare a labeled dataset. Then, the weights of the network are built up, either at random or via transfer learning. As part of the forward propagation process, the network sends out batches of training data and uses the right functions to calculate the loss. To direct future weight updates using optimization methods such as stochastic gradient descent (SGD), backpropagation determines the loss's gradient with regard to the weights. For the purpose of identifying any overfitting, this iterative procedure is repeated for numerous epochs, with close monitoring of the validation loss. B. Tests and Instruction Our Keras-based model employs the Adam optimization technique and cross-entropy loss, starting with a learning rate of 0.0001, in this study. All of the testing and training takes place on a 64 GB RAM-powered, core i9 desktop PC in an

TABLE I ARCHITECTURE SPECIFICATIONS OF THE CNN

Layer	Step size	Output size
Input	/	16384 × 1
Convolutional + ReLu	11 × 1/4	4094 × 96
Maxpooling	3 × 1/2	2046 × 96
Convolutional + ReLu	5 × 1/1	4094 × 128
Maxpooling	3 × 1/2	1022 × 128
Convolutional + ReLu	5 × 1/1	1022 × 192
Convolutional + ReLu	5 × 1/1	1022 × 192
Convolutional + ReLu	5 × 1/1	1022 × 128
Maxpooling	3 × 1/2	510 × 128
Global Average Pooling	/	128
Dense + ReLu	/	128
Dropout(0.5)	/	128
Dense + ReLu	/	64
Dropout(0.5)	/	64
Dense + Softmax	/	4

Anaconda Navigator environment. We train the model using 80% of the samples and evaluate it with 20%. The dataset has 10,000 samples, 1000 samples each (int32 format). Also, as shown in Table I, we use a one-dimensional (1-D) CNN model [10]. Classification accuracy measures such as true positives (TP) and false positives (FP) for each encoder category will be used to evaluate the model's performance.

$$\text{Accuracy(in \%)} = \frac{TP}{TP + FP} \times 100 \qquad (1)$$

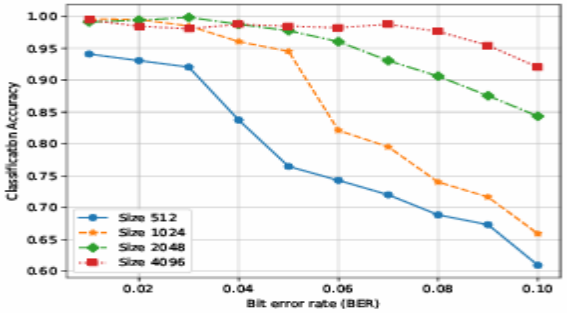

Fig. 3. Classification accuracy of encoders across various BER

SIMULATION RESULTS AND DISCUSSIONS

The Hamming code, with its codeword length of n = 7, code dimension of k = 4, and generator matrix G = [1010011;1001001;0011011;1000101], is used as the block en coder. Also, n= 31 and k= 21 are used in the BCH codification. Our rate-1/2 convolutional encoder has a generator polynomial g = [15,17], a constraint length N0 = 4, and an encoder length of 1/2. A polar encoder, the fourth kind of encoder used, has a reliability sequence of Q = 1024, codeword length N = 16, and message bit length K = 2. We used a separate test dataset to assess the CNN model's efficacy after training the encoder dataset on it.

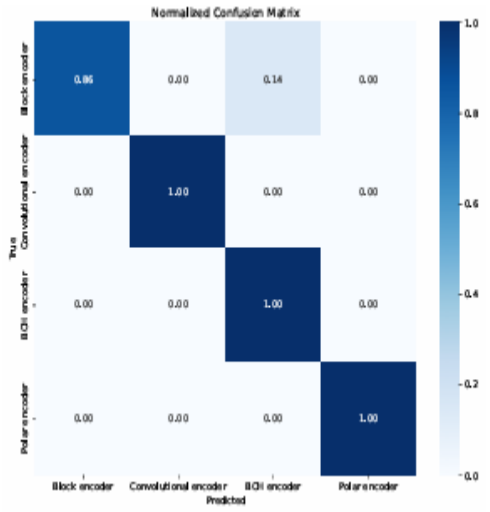

Figure 3 shows the classification accuracy of four different encoders:

polar, convolutional, Hamming, and BCH. With the exception of input size512, the figure shows that the accuracy always exceeds 95% for BER values less than 0.03 and 100% for BER values less than 0.02. The accuracy of the categorization process is significantly improved with higher input sizes. At an input size of 4096 and a BER of 0.02, the confusion matrix for encoder classification is shown in Fig. 4. On one side, we have the predicted values, and on the other, the test results. Squares with a dark blue color represent the encoders' classification accuracy in this matrix. Out of the four encoders, the confusion matrix shows that the Hamming encoder is the only one with a classification accuracy below 85%, while the other three attain a perfect 100%. The reason for this is because compared to other codes, Hamming codes create a very low amount of redundancy. As a consequence, there are fewer different patterns for the CNN to learn.

CONCLUSIONS

To determine which of four possible encoders—block, convolutional, BCH, and polar—was best across an array of inputs, we used a CNN model grounded on deep learning in this work. AWG network. We looked at the correlation between input sample length and classification accuracy and found that it increased as the sample size increased. Importantly, the accuracy always hits 100% for lower BER values before hitting the BER value of 0.02.

REFERENCES

[1]. R. Swaminathan and A. S. Madhukumar, "Classification of error correcting codes and estimation of interleaver parameters in a noisy transmission environment," IEEE Trans. on Broadcast., vol. 63, no. 3, pp. 463-478, Sept. 2017.

[2]. M. Marazin, R. Gautier, and G. Burel, "Dual code method for blind identification of convolutional encoder for cognitive radio receiver design,"in Proc. IEEE GLOBECOM, Honolulu, USA, pp. 1–6, 2009.

[3]. Y. Ding, Z. Huang, and J. Zhou, "An improved blind recognition method for synchronization position and coding Parameters of k/n rate convolutional codes in a noisy environment," IEEE Access, vol. 8, pp. 171305-171315, 2020.

[4]. Y. G. Debessu, H.-C. Wu, and H. Jiang, "Novel blind encoder parameter estimation for turbo Codes," IEEE Commun. Lett., vol. 16, no. 12, pp. 1917–1920, Dec. 2012.

[5]. P. Yu, J. Li, and H. Peng, "A least square method for parameter estimation of RSC sub-codes of turbo codes," IEEE Commun. Lett., vol. 18, no. 4, pp. 644–647, Apr. 2014.

[6]. Swaminathan R, A. S. Madhukumar, N. W. Teck, and S. C. M. Samson, "Parameter estimation of convolutional and helical interleavers in a noisy environment," IEEE Access, vol. 5, pp. 6151-6167, 2017. [7] S. Wee, C. Choi, and J. Jeong, "Novel blind

interleaver parameters estimation based on Hamming weight distribution of linear codes," Digital Signal Process., vol. 117, no. 103190, Oct. 2021.

[7]. J. Wang, C. Tang, H. Huang, H. Wang, and J. Li, "Blind identification of convolutional codes based on deep learning," Digital Signal Process., vol. 115, no. 103086, Aug. 2021.

[8]. H. Song, Y. Chang and K. Fukawa, "Encoding and Decoding of Polar Codes for Frequency Selective Fading Channels," IEEE Vehicular Techno. Conf. (VTC), Helsinki, Finland, pp. 1-5, Aug. 2022.

[9]. J. Wang, C. Tang, H. Huang, H. Wang, and J. Li, "Blind identification of convolutional codes based on deep learning," Digital Signal Process., vol. 115, no. 103086, Aug. 2021.

[10]. A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artif Intell Rev, vol. 53, no. 8, pp. 5455–5516, Apr. 2020