Vol.15, Issue No 2, 2025

Deep Learning Frame Work for Channel Encoders Identification and Categorization using CNN

[1]PALAPARTHI BHARATH SRI SURYA SAI, [2] LINGAMPALLI NAVYA,[3] PERRISETTI LAKSHMI SAI PAVAN, [4] PALA BHARGAV, [5]Mrs. J. PRIYANKA,

[1234]Student, Department of CSE, DNR College of Engineering & Technology, Balusumudi, Bhimavaram, India.

[5]Assistant Professor, Department of CSE, DNR College of Engineering & Technology, Balusumudi, Bhimavaram, India

Abstract

When it comes to digital communication systems, channel encoders are vital for fixing channel-induced random mistakes. In most cases, the receiver has access to details on the transmitting end's channel encoders, including their kind and characteristics. Encoder types and characteristics may be partly or completely unknown in non-cooperative circumstances, such as military communication systems. Four distinct kinds of encoders—block, convolutional, Bose Chaudhuri-Hocquenghem (BCH), and polar—are investigated in this research to see if they can be adequately classified using a deep learning strategy. 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—NCS, deep learning, convolutional neural networks (CNNs), channel encoder classification

INTRODUCTION

Forward error-correcting codes (FEC codes) are essential in digital communication for reducing the impact of random transmitter mistakes [1]. It is essential for decoding to comprehend the receiving end FEC encoders. It is necessary to blindly estimate the channel encoder in certain cases, especially when the receiver does not have previous knowledge of the transmitter's encoder, even if some receiver systems know the transmitter's encoder and can decode successfully [2]. Afterwards, a number of novel algorithms and methods for blindly reconstructing channel encoders have been proposed. But these innovations bring forth fresh difficulties. Solving these problems will have far-reaching consequences, particularly in cases of non-cooperative communication when deciphering messages from unknown sources relies on correct channel encoder reconstruction. Improving spectral efficiency via resource conservation is possible through blind identification of channel encoder settings [1, 2]. Previous research has investigated blind recovery of convolutional encoders using algebraic and dual-code characteristics, as shown in publications like [2] and [3]. 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. In order to identify encoder parameters, the methods shown in [6] used the rank deficit of the data matrix, while the methods presented in [7] used hamming weight distribution. Using the number of non-zero columns and non-zero elements in the column echelon form of the data matrix, the authors presented parameter estimate techniques for Reed-Solomon (RS) codes. Accuracy levels exceeding 90% in blind convolutional code identification using convolutional neural networks (CNNs) have been the subject of recent research. Current approaches are computationally intensive and vulnerable to low signal-to-noise ratios; examples are mathematical algorithms and rank-based methods. On the other hand, our CNN model may not intrinsically comprehend the physics of channel encoding, has trouble remembering long-range dependencies, and struggles with sequential dependencies in channel-encoded data. A. What Came First and What I Did It should be noted that current research has mostly concentrated on applying deep learning approaches to identify individual FEC codes; however, there is a lack of work on classifying channel encoders using CNN. This project's main goal is to use deep learning to classify four channel encoders from an incoming

noisy signal and evaluate the classification accuracy, or the likelihood of correct identification, under different bit error rate (BER) scenarios. For this task, we take into account polar, Bose-Chaudhuri-Hocquenghem (BCH), convolutional, and Hamming encoders.

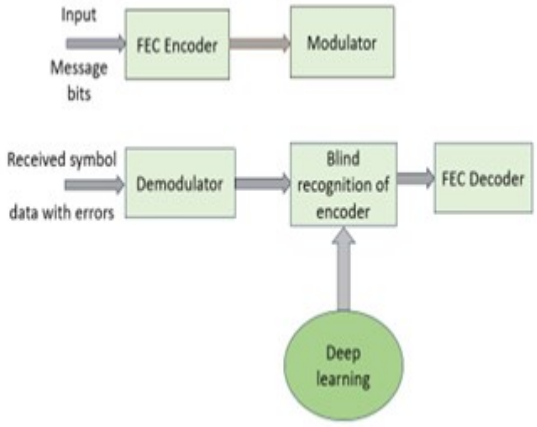

Fig. 1. Encoder classification process

CNN-BASED BLIND ENCODER CLASSIFICATION

In Fig. 1 we can see the FEC encoders being classified. At the outset, the FEC encoder is fed a sequence of bits denoted as b = [b1,b2,...,bk] that are produced at random. An encoder is defined as follows: c = [c1,c2,...,cn], where ci is an element of the Galois Field GF (2). It takes continuous information bits with a block size of k and generates an encoded data bit sequence with a block size of n. 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. At the transmitter, the chosen modulation techniques incorporate binary phase-shift keying (BPSK). The data bits, denoted as y = [y1,y2,...,yn], are extracted from the received signal after demodulation at the receiving end. Subsequently, the data bits are sent on to the FEC decoder to recover the initial information bits. The main goal of this study is to develop a CNN model that can autonomously identify or classify encoders. There are many digital communication and storage systems that make use of the channel encoders under consideration. These systems include satellite communications, Wi-Fi, and mobile communication standards such as GSM, CDMA, and 5G new-radio.

Part A: Creating a Database We use MATLAB to generate datasets for four distinct coding methods in our experimental setting. 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. The foundational principle of polar codes is channel polarization; they were the first codes that provide direct evidence of 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]. On the other hand, block codes divide data into fixed-size chunks and process 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. Contrarily, convolutional encoders work with a continuous data stream, which is why they differ from block codes. It takes bit-by-bit input data and processes it in a shift register, then uses modulo-2 additions to combine the contents of the registers to produce the encoded output. Standard notation for block codes is (n,k), where r = n−k represents the length of the parity or redundancy bits. The length of the coding constraint is expressed by N0 in the convolutional code described in this study, which is represented by (n,k,N0). In this study, the polar codes are represented by the variables (N,K). Here, N is the codeword length, which is always an exponent of 2 (i.e., 2n for all n ≥ 2), and K is the variable that represents

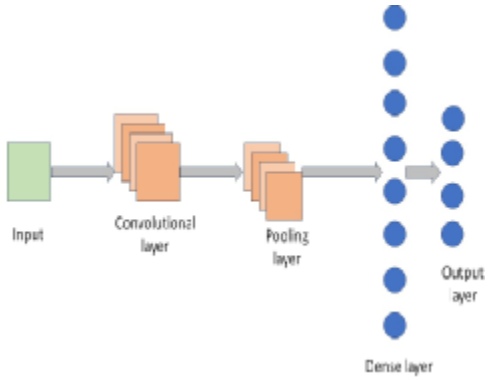

Fig. 2. Basic structure of CNN the number of message bits.

PROPOSED CNN 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. Images or data structured like a grid may be inputted 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. The following pooling layers keep important information while reducing the spatial dimensions of the feature maps. 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—Learning It is necessary to initialize weights, perform forward propagation, calculate losses, backpropagation, and weight up dates in order to train a convolutional neural network (CNN). Gathering and preprocessing a labeled dataset is the first step, followed by randomized or transfer learning-based weight initialization of the network. Forward propagation involves sending training data in batches across the network and then applying the right functions to calculate the loss. After then, optimization methods like stochastic gradient descent (SGD) are used to guide further weight updates by calculating the loss gradients with respect to the weights via backpropagation. To identify any overfitting, this iterative procedure is performed for several epochs, and keeping an eye on the validation loss is critical. B. Instruction and Evaluation Our Keras-based model employs the Adam optimization technique and cross-entropy loss, starting with a learning rate of 0.0001, in this study. A desktop PC with 64 GB of RAM and a core i9 CPU is used for both training and testing in an

TABLE I Block encoder COMSNETS 2024 - Poster Track Normalized Confusion Matrix ARCHITECTURE SPECIFICATIONS OF THE CNN

Layer	Step size	Output size
Input	/	16384×1
Convolutional + ReLu	$11 \times 1/4$	4094×96
Maxpooling	$3 \times 1/2$	2046×96
Convolutional + ReLu	$5 \times 1/1$	4094×128
Maxpooling	$3 \times 1/2$	1022×128
Convolutional + ReLu	$5 \times 1/1$	1022×192
Convolutional + ReLu	$5 \times 1/1$	1022×192
Convolutional + ReLu	$5 \times 1/1$	1022×128
Maxpooling	$3 \times 1/2$	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. of 80% set aside for training the model and 20% for assessment, we produce a dataset of 10,000 samples, each 1000 in length (int32 format). We also make use of a one-dimensional (1-D) CNN model, the specifics of which are laid forth in Table I [10]. Our objective is to evaluate the model's performance by determining its classification accuracy for each encoder category using metrics such as true positives (TP) and false positives (FP) as

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

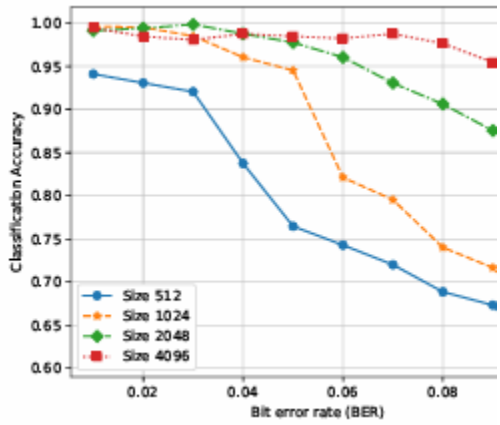

Fig. 3. Classification accuracy of encoders across various BER

SIMULATION RESULTS AND DISCUSSIONS

(1) 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. Plus, we use BCH coding with n= 31 and k= 21. We take into account a convolutional encoder with a rate of 1/2, a generator polynomial g = [15,17], and a constraint length N0 = 4. As for the fourth encoder, it's a polar encoder that uses a Q = 1024 reliability sequence, codeword length N = 16, and message bit length K = 2. After training the CNN model on the encoder dataset, we used a specialized test to assess its performance.

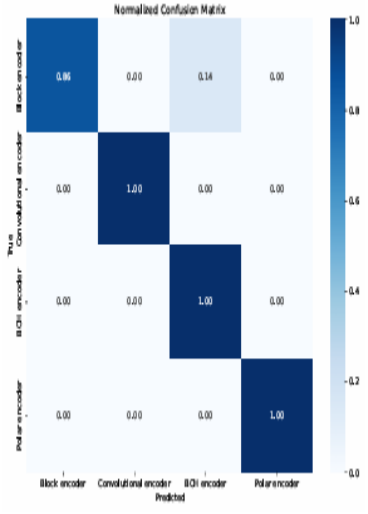

Fig. 4. Confusion matrix for the classification accuracy of encoders at 0.02 BER value

data set. Figure 3 shows the classification accuracy of four different encoders: polar, convolutional, Hamming, and BCH. As shown in the picture, the accuracy always exceeds 95% for BER values below 0.03 and 100% for BER values less than 0.02 (with the exception of input size 512), regardless of the input length. The accuracy of the categorization process is significantly improved with higher input sizes. The confusion matrix for encoder classification with 4096 input size and a BER of 0.02 is shown in Figure 4. The horizontal axis represents test results, whereas the vertical axis represents predicted values. 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%. This is due to the fact that, in comparison to other codes, Hamming codes provide somewhat less redundancy; hence, there are fewer different patterns for the CNN to learn due to this reduced redundancy.

Conclusion

Final Thoughts In this paper, we used a deep learning convolutional neural network (CNN) model to determine which of four encoders—block, convolutional, BCH, and polar—should be used over an AWGN channel. We looked at the correlation between input sample length and accuracy in classification, and we found that it increased as the sample size increased. The accuracy always hits 100% at lower BER values before hitting the 0.02% BER threshold, which is noteworthy.

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.

[8]. 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.

[9]. 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.

[10]. 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.

[11]. 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.