Classification of Channel Encoders Using Convolutional Neural Network

Nayim Ahamed, Naveen B, and Swaminathan R.
Department of Electrical Engineering, Indian Institute of Technology (IIT) Indore
Email: phd22012020@iiti.ac.in, mt2102102010@iiti.ac.in, swamiramabadran@iiti.ac.in

2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 979-8-3503-8311-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/COMSNETS59351.2024.10427098

***Abstract*—In digital communication systems, channel encoders play a crucial role in rectifying random errors introduced by the channel. Typically, information about the type and parameters of channel encoders used at the transmitting end is available at the receiver. However, in non-cooperative scenarios like military communication systems, encoder types and parameters may be only partially known or entirely unknown. This paper explores the feasibility of employing a deep learning approach to classify four different types of encoders: block, convolutional, Bose-Chaudhuri-Hocquenghem (BCH), and polar encoders. Utilizing a convolutional neural network (CNN) model for classification, our proposed approach achieves classification accuracy exceeding 95% upto bit-error-rate (BER) value of 0.03. The results also indicate that the accuracy improves with the input sample length.**

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

I. Introduction

In digital communication, forward error-correcting (FEC) codes play a pivotal role in mitigating random errors at the transmitter [1]. Understanding FEC encoders at the receiving end is crucial for decoding. While some receiver systems possess prior knowledge of the encoder used at the transmitter, enabling successful decoding, situations arise where blind estimation of the channel encoder becomes imperative, particularly when the receiver lacks prior knowledge [2]. Consequently, various innovative techniques and algorithms for the blind reconstruction of channel encoders have been introduced. However, these advancements pose new challenges. Addressing these challenges have significant implications, especially in non-cooperative communication scenarios, where accurate reconstruction of the channel encoder is vital for successfully decoding the signals from unknown sources. Blind recognition of channel encoder parameters can enhance spectral efficiency by conserving channel resources [1], [2]. The blind recovery of convolutional encoders has been explored in prior research, as seen in works such as [2] and [3], leveraging algebraic and dual-code properties. Studies in [4] and [5] delved into the blind reconstruction of a parallel concatenation of two recursive systematic convolutional (RSC) encoders using iterative expectation-maximization (EM) and the least-square method. Methods in [6] were based on the rank deficiency of the data matrix for recognizing encoders parameters, while [7] employed hamming weight distribution for the same purpose. The authors introduced parameter estimation algorithms for Reed-Solomon (RS) codes, leveraging the count of non-zero columns and non-zero elements in the column echelon form of the data matrix. Recent literature explored blind identification of convolutional codes based on convolutional neural network (CNN) with accuracy above 90 %. Existing models, such as rank-based and mathematical algorithmic models, are sensitive to low signal-to-noise ratios and computationally complex. In contrast, our CNN model faces challenges with sequential dependencies in channel-encoded data, lacks memory for long-range dependencies, and may not inherently understand the physics of channel encoding.

A. Motivations and Contributions

Notably, while existing research has focused on employing deep learning techniques for recognizing individual FEC codes, there is a gap in addressing the classification of channel encoders using CNN. The principal objective of the proposed work is to leverage deep learning techniques for the classification of four different channel encoders from the incoming noisy signal and assess the classification accuracy, specifically the probability of correct identification, under varying bit error rate (BER) conditions. We consider Hamming, convolutional, Bose-Chaudhuri-Hocquenghem (BCH), and polar encoders for the classification process.

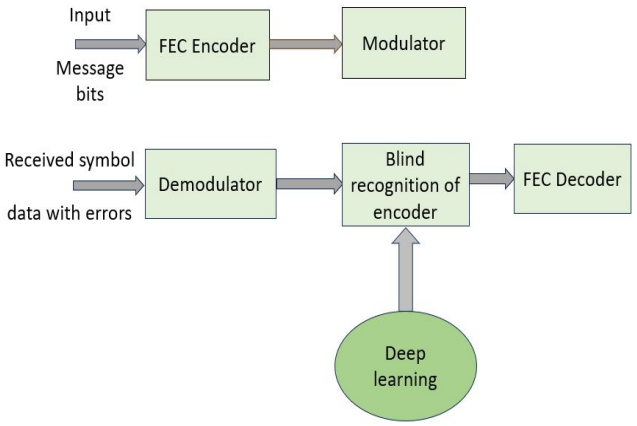

Fig. 1. Encoder classification process

II. CNN-based blind encoder classification

The process of classifying the FEC encoders is illustrated in Fig. 1. Initially, a series of randomly generated information bits

represented as $b = [b_1, b_2, ..., b_k]$ is introduced into the FEC encoder. This encoder manages the continuous information bits with a block size of k and produces an encoded data bit sequence with a block size of n, denoted as $c = [c_1, c_2, ..., c_n]$, where each c_i is an element of the Galois Field GF(2). The code rate is given by $r = k/n$, where k and n denote code dimension and codeword length, respectively. After encoding, the resultant digital sequence undergoes modulation before transmission through the communication channel. The selected modulation schemes include binary phase-shift keying (BPSK) at the transmitter. Upon arrival at the receiving terminal, the received signal undergoes demodulation to extract the data bits, represented as $y = [y_1, y_2, ..., y_n]$. After that the data bits are then transmitted to the FEC decoder for the retrieval of the original information bits. This research predominantly focuses on the autonomous identification or classification of encoders using a CNN model. The channel encoders under consideration find extensive applications in digital communication and storage systems, including satellite communications, wireless fidelity (Wi-Fi), and mobile communication standards like global system for mobile communications (GSM), code division multiple access (CDMA), and 5G new-radio (NR).

A. Dataset Generation

In our experimental setup, MATLAB is employed to generate datasets for four different coding schemes. We assume successful demodulation of information signal at the receiving end, along with perfect frame synchronization. Consequently, we assume additive white Gaussian noise (AWGN) channel. The paper features four FEC codes, including polar code. Polar codes, being the first codes with explicit proof of achieving channel capacity for symmetric binary-input, discrete, memoryless channels (B-DMC), are based on the concept of channel polarization. The reliability sequence is crucial for both encoding and decoding, defined using a generator matrix established recursively through the Kronecker product [9].

In contrast, block codes operate by segmenting data into fixed-size blocks, treating each independently. For our experiment, the choice is a Hamming block code, often applied for single-error correction. In addition, BCH codes, also falling under the category of block codes, are renowned for their robust error correction capabilities, proficient in addressing both random and burst errors across diverse applications. On the other hand, convolutional encoder deviates from block codes as it operates on a continuous data stream. It utilizes a shift register to process input data bit by bit, combining shift register contents through modulo-2 additions to generate the encoded output.

Block codes are typically denoted as (n, k), where $r = n-k$ denotes the length of the parity or redundancy bits. The convolutional code described in this paper is denoted by (n, k, N_0), where N_0 represents the length of the coding constraint. The polar codes described in this paper are denoted by (N, K), where N represents the codeword length, always taking the form of an exponent of 2, i.e. 2^n $(n \geq 2)$, and K represents the number of message bits. The difference $N-K$ corresponds to the number of frozen bits with reliability sequence Q [9].

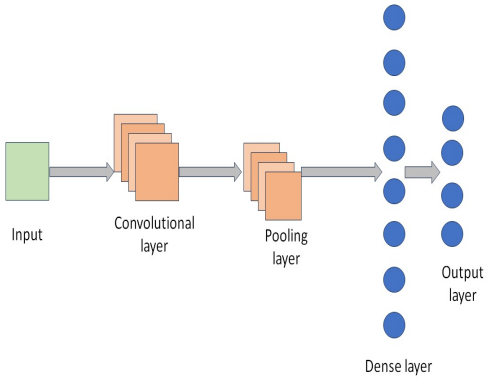

Fig. 2. Basic structure of CNN

III. Proposed CNN model

A CNN architecture consists of key components [10], including an input layer, an output layer, and multiple layers for convolution, pooling, and fully-connected operations, as illustrated in Fig. 2. The input layer is designed to receive image or grid-like data. The core elements of a CNN are the convolutional layers, which employ multiple filters to extract features [11]. These filters conduct convolution operations across the input data, identifying patterns such as edges, textures, and basic shapes. Filters, also referred to as kernels, are small-sized matrices or tensors that convolve across the input data during the operation of convolutional layers. In the subsequent pooling layers, spatial dimensions of the feature maps are reduced while retaining crucial information. This reduction is achieved through commonly used pooling techniques like max-pooling and average-pooling. Finally, the features extracted from the previous layers are processed beyond the capabilities of fully-connected networks. This is accomplished through a softmax-based probability distribution.

A. CNN network learning

The training of a CNN involves initializing weights, forward propagation, loss calculation, backpropagation, and weight updates. Initially, a labeled dataset is collected and preprocessed, and the network's weights are initialized, either randomly or through transfer learning. During forward propagation, batches of training data traverse the network, and the loss is computed using appropriate functions. Backpropagation then calculates gradients of the loss with respect to the weights, guiding subsequent weight updates using optimization algorithms like stochastic gradient descent (SGD). This iterative process is repeated for multiple epochs and monitoring the validation loss is crucial for detecting potential overfitting.

B. Training and Testing

In this work, our Keras-based model uses cross-entropy loss and the Adam optimization algorithm with initial learning rate equal to 0.0001. Training and testing occur on a personal computer with 64 GB RAM and a core i9 processor in an

TABLE I
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. We generate a dataset of 10,000 samples, each 1000 in length (int32 format), allocating 80% for model training and 20% for evaluation. Further, we use one-dimensional (1-D) CNN model [10] and the specific details are outlined in Table I. Our goal is to assess the model's performance by calculating its classification accuracy using metrics like true positives (TP) and false positives (FP) for each encoder category as

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

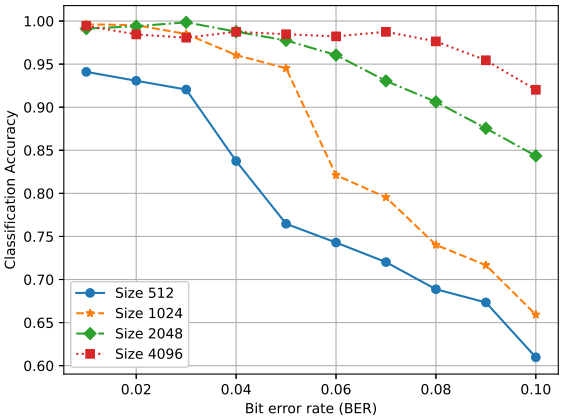

Fig. 3. Classification accuracy of encoders across various BER

IV. Simulation Results and Discussions

We adopt the Hamming code as the block encoder, characterized by a codeword length of $n = 7$, code dimension $k = 4$, and generator matrix $G = [1010011; 1001001; 0011011; 1000101]$. Further, BCH code is employed with $n = 31$ and $k = 21$. We consider a rate-1/2 convolutional encoder with constraint length $N_0 = 4$ and generator polynomial $g = [15, 17]$. The fourth encoder utilized is a polar encoder with a reliability sequence of $Q = 1024$ for codeword length $N = 16$ and length of message bit $K = 2$.

Following the training of the encoder dataset on the CNN model, we evaluated its performance using a dedicated test dataset. The classification accuracy of four distinct encoders (i.e. Hamming, BCH, convolutional, and polar) is illustrated in Fig. 3. The accuracy, as depicted in the figure, consistently surpasses 95% across varying input lengths for BER values below 0.03 and 100% for BER values less than 0.02, except for the input size 512. Notably, larger input sizes exhibit enhanced accuracy in the classification process. Fig. 4 presents the confusion matrix for encoder classification at an input size of 4096 and a BER of 0.02. Predicted values are represented along the vertical axis, while test values are along the horizontal axis. In this matrix, dark blue squares signify the classification accuracy of encoders. The confusion matrix reveals that only the Hamming encoder exhibits an accuracy of 86%, whereas the remaining three encoders achieve a perfect 100% classification accuracy. This is because, Hamming codes introduce a relatively lower level of redundancy compared to other codes; hence, this lower redundancy results in fewer distinct patterns for the CNN to learn.

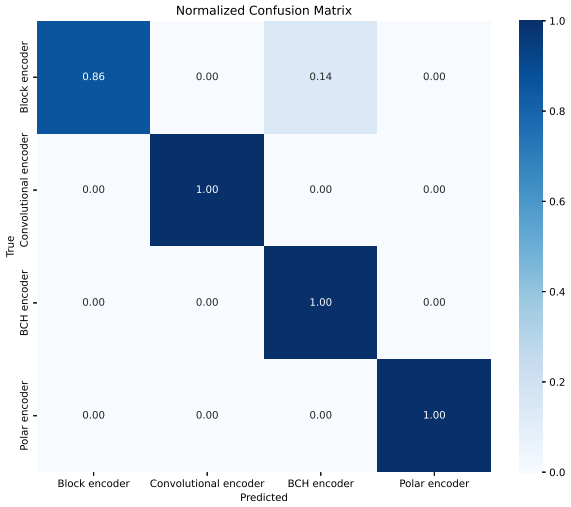

Fig. 4. Confusion matrix for the classification accuracy of encoders at 0.02 BER value

V. Conclusions

In this manuscript, we utilized a CNN model based on deep learning for the purpose of identifying the correct encoder among block, convolutional, BCH, and polar encoders over an AWGN channel. Our analysis of classification accuracy considered varying input sample sizes, revealing that an increase in the input sample length correlates with higher accuracy. Notably, at lower BER, the accuracy consistently achieves 100% before reaching the BER value of 0.02.

Acknowledgment

This work is supported by the Visvesvaraya PhD Scheme for Electronics & IT (MeiTy) and EMR-II scheme of CSIR-HRDG, Govt. of India (Project no. 22/0881/23/EMR-II).

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.