A Novel Iterative Decoding for Iterated Codes Using Classical and Convolutional Neural Networks

Marek Blok[1][0000-0002-8793-1697] and Bartosz Czaplewski[2][0000-0001-7904-5567]

[1] Artificial Intelligence Center of Excellence, Hapag-Lloyd Knowledge Center Sp. z o.o., Grunwaldzka 413, 80-309 Gdańsk, Poland
[2] Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, Gabriela Narutowicza 11/12, 80-233 Gdansk, Poland
bartosz.czaplewski@pg.edu.pl

Abstract. Forward error correction is crucial for communication, enabling error rate or required SNR reduction. Longer codes improve correction ratio. Iterated codes offer a solution for constructing long codes with a simple coder and decoder. However, a basic iterative code decoder cannot fully exploit the code's potential, as some error patterns within its correction capacity remain uncorrected. We propose two neural network-assisted decoders: one based on a classical neural network, and the second employing a convolutional neural network. Based on conducted research, we proposed an iterative neural network-based decoder. The resulting decoder demonstrated significantly improved overall performance, exceeding that of the classical decoder, proving the efficient application of neural networks in iterative code decoding.

Keywords: Forward Error Correction, Neural Networks, Iterated Codes.

1 Introduction

Forward error correction (FEC) is an effective approach to error detection and correction. This protection is achieved by adding parity bits to the transmitted data. Longer codes offer better protection but typically require more complicated decoders. An interesting method of long code construction with simple decoder are iterated codes [1,2]. They combine two or more simpler codes into one longer code. Using this approach, a message can be fit into 2D table and then encoded iteratively, i.e. the rows are encoded first, followed by the encoding of the columns. Decoding is a reverse process and is also iterative. The correction capacity of the iterated code significantly increases since the minimum Hamming distance of the code is a product of the distances of the component codes. Unfortunately, the classic, iterative, decoder is not able to correct all error patterns even if their weight is within the code's correction capacity. To address this, we explore the error correction capabilities of classical neural networks (NN) and convolutional neural networks (CNN) [3-7].

This paper aims to introduce an innovative decoder structure for iterated codes, utilizing NNs and CNNs to improve error correction effectiveness. To the best of the authors' knowledge, this paper represents the first published research on decoding

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

iterated codes using machine learning techniques ([8] was the groundwork). The contribution comprises three aspects: 1. custom datasets; 2. NN and CNN models trained for the considered iterated code; 3. NN-based and CNN-based iterated decoder, wherein the proposed NN and CNN are used iteratively. The CNN-based iterative decoder achieves higher error correction efficiency than the classic detector.

2 Related Work

There are publications that use machine learning for decoding different FEC codes: Hamming codes, Bose–Chaudhuri–Hocquenghem (BCH) codes, polar codes, turbo codes, Reed–Solomon (RS) codes, or Low-Density Parity-Check (LDPC) codes.

Article [9] introduced an RNN-based decoder for Hamming codes, offering the advantage of correcting noisy code words in non-binary form. In [10], the BCH codes were decoded with recurrent neural network (RNN). The interconnection of decoder parameters in iterations to form a RNN was implemented. The article [11] involves incorporating redundant symbols for synchronization in noisy channels. It enhances decoding precision by forwarding soft decisions from the synchronization module to the receiver's decoder. The next article [12] introduced successive cancellation and belief propagation decoding algorithms for polar codes. It discusses deep learning (DL) decoders, their principles and performance assessments. In [13] an architecture for turbo code decoding was proposed. This Deep Turbo decoder showed outstanding performance in both AWGN (Additive White Gaussian Noise) and non-AWGN channels. In [14], DNNs for improving belief propagation decoding were tested for BCH codes with cycle-reduced parity-check matrices. It provided guidelines for refining a model by incorporating domain knowledge into a DL model. In [15], the authors used the advantages of DL and conventional LDPC decoding with normalized min-sum by distributing the iterative decoding between check nodes and variable nodes in a forward propagation network. In [16], RNN-based decoder was tested on recursive systematic convolutional (RSC) codes. In [17], graph neural network (GNN) was utilized for LDPC and BCH codes. Unlike many DL-based decoding methods, that solution is scalable to arbitrary block lengths and minimalizes problem of dimensionality during training. Next study [18] explored the integration of NN into polar codes decoding. The authors showed a fresh approach to reducing complexity by leveraging constituent codes and introduced a scheduling scheme for decoding latency reduction. The recent work [19] focused on a CNN-based LDPC decoder tailored for optical fiber communication systems. Employing a correlated Gaussian noise model CNN can identify noise correlation, leading to enhanced decoding performance through iterative processing. The study [20] introduced shared graph neural network (SGNN) decoding algorithms, reducing parameters by half or three-quarters for BCH and LDPC codes, while maintaining a marginal degradation in performance.

The solutions proposed in this article stem from research initiated in [8], which first attempted to use NNs to decode iterated codes. The article presented two NNs trained on different datasets for error detection and correction, and an NN-assisted decoder for iterated codes leveraging both networks to decode iterated codes.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

3 The Considered Iterated Code

Let's consider double-iterated code [1,2] composed of two FEC codes capable of correcting a single error: a Hamming code (7,4) with the generator matrix $\mathbf{G}_1$ and a shortened Hamming code (6,3) with the generator matrix $\mathbf{G}_2$ (2). With $\mathbf{G}_i$ and information vector $\mathbf{x}$, a code word $\mathbf{c}_i$ for the i-th component code is obtained as in (1).

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{1}$$

$$\mathbf{c}_i = \mathbf{x} \cdot \mathbf{G}_i \tag{2}$$

The first code has 4 information bits and a length of 7, while the second has 3 information bits and a length of 6. We split the message into 12-bit information blocks formed as a 3x4 matrix. First, each row is encoded with first code $\mathbf{G}_1$ resulting in 3 rows of 7-bit words. Then, each column is encoded with the second code $\mathbf{G}_2$ resulting in 7 columns of 6-bit words. The resulting 6x7 matrix forms a 42-bit code word with 12 information bits. The code rate is 0.285, and the error-correcting capacity is 4.

The classical decoder also follows the iterated approach of the coder. The parity check matrices (3) of Hamming codes can be easily obtained from generator matrices.

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{s}_i = \mathbf{H}_i \cdot \mathbf{y} \tag{4}$$

The received vector $\mathbf{y}$ is converted into 6x7 matrix and the decoder calculates the syndrome $\mathbf{s}_i$ for each row with $\mathbf{H}_1$ and each column with $\mathbf{H}_2$ as in (4). For both component codes, syndromes are binary vectors of length 3. A syndrome with all zeros means that there is no need for correction while the other possible values correspond to 7 error patterns that are used to correct errors in rows and columns accordingly.

4 Network Structures

The first concept was to use a NN. Since syndrome calculations and error correction equations used in the studied iterated code are not highly complex mathematical problems, it was initially assumed that such a simple solution would suffice. Various NN structures with different numbers of layers, neurons, and activation functions were examined. After a grid search, the structure depicted in Fig. 1 exhibited the best results. The proposed NN consists of an 1x42 input layer for a 42-bit received message, two fully connected layers (512 neurons each), and an 1x42 output layer for a

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

42-bit predicted error vector. ReLU activation functions are applied after the hidden layers, while a sigmoid activation function is used after the output layer. The total number of trainable parameters is 306218.

The second concept was to use a CNN that utilizes spatial properties of the code. The encoding of a 6x7 matrix is done by solving 2D parity-check equations which impact individual rows and columns. In the result, an error in a received message distorts only parity checks in the row and the column at which it is located. It was hypothesized that it is possible to use CNN to learn these spatial features using filters in the sizes of the rows and columns in the message matrix. Various CNN structures were examined in a grid search. The structure presented in Fig. 2 exhibited the best results. Here, the input layer size is 6x7 for a 42-bit received message. The data is passed to two parallel convolutional layers: one for scanning the rows of the message (512 horizontal filters, size of 1x7, ReLU activation) and the other for scanning the columns (512 vertical filters, size of 6x1, ReLU activation). No pooling layers were used due to the low data dimensionality and no dropout layers were employed because overfitting is not a concern given the nature of the data. Convolutional layers produce 512 6x1 and 512 1x7 feature maps, which are concatenated into a feature vector of length 6656 and passed to two fully connected layers (512 and 256 neurons, ReLU activation). The output layer utilizes a sigmoid activation function and outputs a 42-bit predicted error vector. The number of trainable parameters is 3558186.

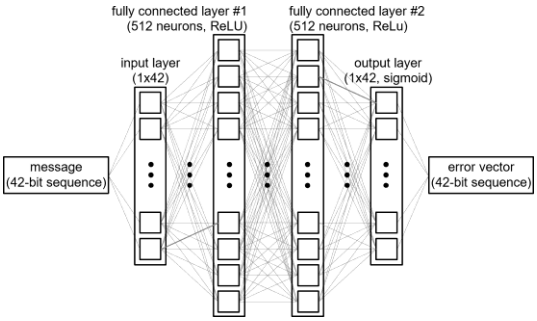

Fig. 1. The proposed NN model.

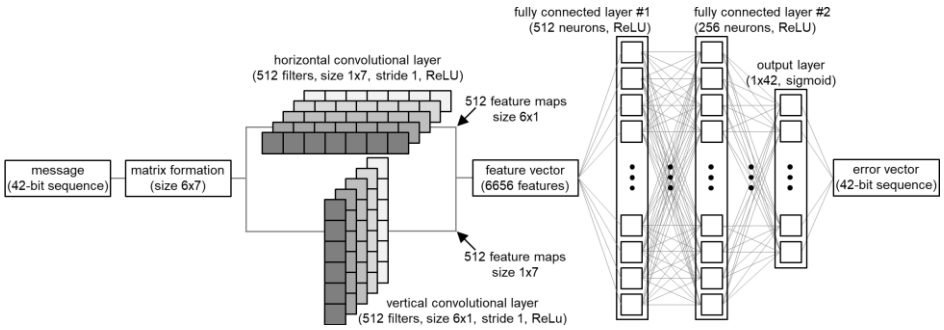

Fig. 2. The proposed CNN model.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

5 Network Training

The training dataset consists of 872086 pairs of 42-bit messages and 42-bit error vectors. The number of errors in messages in the training dataset ranges from 1 to 4. This way, the networks can learn the actual behavior of the code because the code's error-correcting capability is equal to 4. The training dataset was utilized for supervised learning through a 10-fold cross-validation approach. Average validation loss and accuracy were calculated for the entire 10-fold cross-validation. The model with the lowest validation loss among all the folds was selected for the testing phase.

The test dataset contains 4912 pairs of 42-bit messages and 42-bit error vectors. The number of errors in messages in the test dataset ranges from 0 to 5. This way, we can verify the model's ability to predict zero error vectors or vectors with error weight exceeding the error-correcting capability of the code The test dataset was used for the final verification of the trained model. Total test loss, total test accuracy, and test accuracy for specific error weights were computed.

Learning parameters were assessed using the grid search method with the criterion of effective network training within a reasonable training time. Batch size was 128, solving algorithm was Adam (beta1=0.9, beta2=0.999, eps=1e-08, decay=0), initial learning rate was 0.001, learning rate schedule was ReduceLROnPlateau (mode='min', factor=0.1), loss function was binary cross entropy (BCEWithLogitsLoss from the PyTorch), and max epochs was 200 with validation patience of 2. The implementation environment used Python 3.11.0, PyTorch 2.0.1+cu118, Nvidia GeForce GTX 1080 Ti GPU, and CUDA 12.2.

6 The Proposed Neural Network-based Iterative Decoder

To analyze the networks' performance in more detail, the classification accuracy was calculated additionally for datasets with messages containing from 0 up to 7 errors. For weights 0 and 1 all possible combinations of code words and error patterns have been tested (4096 and 172033 messages) while for larger weights 200000 randomly selected unique messages have been used in tests. We analyzed the error correction patterns for the classic decoder and the proposed networks as stand-alone detectors. Due to the limited number of pages, we are unable to include these results.

It was observed that the classic iterated decoder does not correct all the errors (misses 1, 2, or 4 errors) and at the same time adds additional errors (up to 3 new errors) when decoding messages with error vectors of weight 4. This pattern is different for NN and CNN networks, which are more likely to omit or add to the message just a single error, and almost never incorrectly detect all 4 errors. The neural networks working as stand-alone detectors almost always manage to detect some of the errors from the pattern, and in most cases when they fail and introduce additional errors, they add just one error. Thus, even if the proposed networks fail to correctly decode messages, the error pattern weight is decreased; thus, it is very likely that the network will be able to correct the message in the second attempt. These observations led us to the proposed neural network-based iterative decoder presented in Fig. 3.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

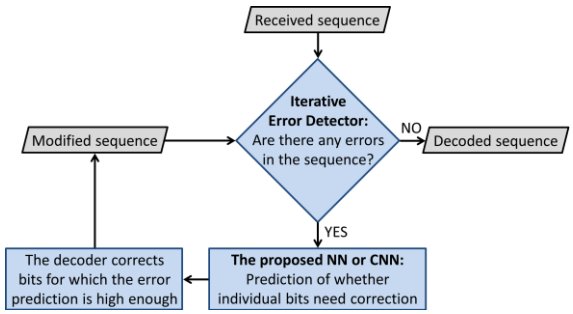

Fig. 3. The proposed neural network-based iterative decoder.

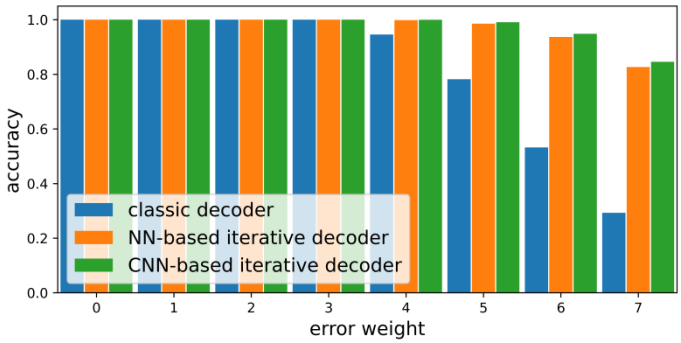

Fig. 4. Test accuracy results of the proposed neural network-based iterative decoders.

In Fig. 4 the iterative neural network-based decoders are directly compared with the classic decoder. The number of correction attempts in the experiments has been limited to 9. As we can see in Fig. 4, both NN and CNN architectures used iteratively significantly improve their performance with CNN achieving better results.

As we can see in Table 1, the NN-based iterative decoder almost perfectly corrected messages with error weights equal up to 3, only marginally being worse than the classic decoder in those cases, while beating it for larger error weights. These results have been even improved by the CNN-based iterative decoder which corrected all errors with the weight up to 3, provided almost perfect results for weights 4 and 5, and even showed very high performance for error vector weights of 6 and 7. Compared to the classic decoder, the results for the proposed CNN-based iterative decoder are a major improvement over a classical approach, especially in cases where the weight of the error vector exceeds the correction capacity of the code.

In most cases, the first attempt leads to the correct solution. Nevertheless, in the remaining cases, additional correction attempts are required. The distributions of the numbers of iterations for different error weights are presented in Table 2. The second correction attempt is usually sufficient with additional iterations needed for errors weights above 4. It can be observed that the CNN-based iterative decoder not only allows for improved accuracy but also reduces the number of required iterations.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

Table 1. Test accuracy results of the iterative decoders with specific error vectors' weights.

dataset error weight	classic decoder	NN-based decoder	CNN-based decoder
0-2	1.00000	1.0	1.0
3	1.00000	0.999998	1.0
4	0.94565	0.998591	0.999861
5	0.78226	0.985696	0.990726
6	0.53211	0.936987	0.948255
7	0.29275	0.826528	0.845893

Table 2. Percentage of cases of decoding finished at a given iteration.

Number of iterations	1	2	3	4	5	6+
NN-based iterative decoder	69,36%	24,67%	4,43%	1,02%	0,25%	0,27%
CNN-based iterative decoder	71,70%	24,70%	3,17%	0,36%	0,04%	0,03%

7 Conclusions

It is possible to train a neural network to decode iterated codes. The analyzed problem relies on the network's ability to learn decoding rules from limited examples. The proposed networks, even if they fail to correct messages, typically decrease the number of errors. The proposed NN-based and CNN-based iterative decoders outperform the classic code decoder, with CNN-based iterative decoder proving particularly effective. For this study, we have selected an iterated code based on two different component codes, with a decoder not fully exploiting the code's capacity. These results prompt further research on different iterated codes or a wider range of FEC codes.

Acknowledgments. This study was funded by the statutory activities of Faculty of Electronics, Telecommunications, and Informatics of Gdańsk University of Technology, and by the research subsidy from the Polish Ministry of Science and Higher Education.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Elias, P.: Error free coding. IRE Trans. Information Theory, PGIT-4, 29-37 (1954)
2. Lin, S., Costello, D.: Error Control Coding, 2nd edition, Prentice Hall (2004)
3. Aggarwal, C.C.: Neural Networks and Deep Learning: A Textbook. Springer: Cham (2019)
4. Krohn, J., Beyleveld, G., Bassens, A.: Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence. Addison-Wesley Data & Analytics Series, Addison-Wesley (2019)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28

5. Czaplewski, B., Dzwonkowski, M.: A novel approach exploiting properties of convolutional neural networks for vessel movement anomaly detection and classification. ISA Transactions **119**, 1–16 (2022)
6. Czaplewski, B., Dzwonkowski, M., Panas, D.: Convolutional Neural Networks for C. Elegans Muscle Age Classification Using Only Self-Learned Features. Journal of Telecommunications and Information Technology **4** (2022)
7. Czaplewski, B.: An Improved Convolutional Neural Network for Steganalysis in the Scenario of Reuse of the Stego-Key. In: Tetko I., Kůrková V., Karpov P., Theis F. (eds.) Artificial Neural Networks and Machine Learning – ICANN 2019: Image Processing. ICANN 2019. Lecture Notes in Computer Science, vol 11729. Springer, Cham (2019)
8. Fiedot, O., Blok, M.: Decoding of iterated codes with the use of a neural network. In Proceedings of the KRiT 2023 Conference on Radiocommunications and Teleinformatics, Cracow, Poland (2023)
9. Abdelbaki, H., Gelenbe, E., El-Khamy, S.E.: Random Neural Network Decoder for Error Correcting Codes. In Proceedings of the International Joint Conference on Neural Networks, vol. 5, pp. 3241-3245 (1999)
10. Nachmani, E., Marciano, E., Lugosch L., Gross, W.J., Burshtein, D., Be'ery, Y.: Deep Learning Methods for Improved Decoding of Linear Codes. IEEE Journal of Selected Topics in Signal Processing **12**(1), 119–131 (2018)
11. Chadov, T.A., Erokhin, S.D., Tikhonyuk, A.I.: Machine learning approach on synchronization for FEC enabled channels. In Proceeding of 2018 Systems of Signal Synchronization, Generating and Processing in Telecommunications, Minsk, Belarus (2018)
12. Gross, W.J., Doan, N., Mambou, E.N., Hashemi, S.A.: Deep Learning Techniques for Decoding Polar Codes. In: Machine Learning for Future Wireless Communications, Luo, F.-L. (eds), John Wiley & Sons Ltd. (2020)
13. Jiang, Y., Kim, H., Asnani, H., Kannan, S., Oh, S., Viswanath, P.: DEEPTURBO: Deep Turbo Decoder. In Proceedings of the IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications SPAWC (2019)
14. Be'ery, I., Raviv, N., Raviv, T., Be'ery, Y.: Active Deep Decoding of Linear Codes. IEEE Transactions on Communications **68**(2), (2020)
15. Wang, Q., Wang, S., Fang, H., Chen, L., Chen, L., Guo, Y.: A Model-Driven Deep Learning Method for Normalized Min-Sum LDPC Decoding. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1-6, Dublin, Ireland (2020)
16. Devamane, S.B., Itagi, R.L.: Recurrent neural network based turbo decoding algorithms for different code rates. Journal of King Saud University – Computer and Information Sciences **34**, 2666–2679 (2022)
17. Cammerer, S., Hoydis, J., Aoudia, F.A., Keller, A.: Graph Neural Networks for Channel Decoding. In Proceedings of 2022 IEEE Globecom Workshops (GC Wkshps), pp. 486-491, Rio de Janeiro, Brazil (2022)
18. Zhou, L., Zhang, M., Chan, S., Kim, S.: Review and Evaluation of Belief Propagation Decoders for Polar Codes. Symmetry **14**(12), 2633 (2022)
19. Zhang, J., Jiang, W., Zhou, J., Zhao, X., Huang, X., Yu, Z., Yi, X., Qiu, K.: An iterative BP-CNN decoder for optical fiber communication systems. Optics Letters **48**(9), 2289-2292 (2023)
20. Wu, Q., Ng, B.K., Lam, C.-T., Cen, X., Liang, Y., Ma, Y.: Shared Graph Neural Network for Channel Decoding. Appl. Sci. **13**(23), 12657 (2023)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_28