High-Efficient Reed-Solomon Decoder Based On Deep Learning

Xiangyu An, Yu Liang and Wei Zhang
School of Microelectronics
Tianjin University
Tianjin 300072, China
Email:{anxiangyu, liangyu, tjuzhangwei}@tju.edu.cn

***Abstract*—Deep learning recently shows outstanding potential in channel decoding optimization, but its effect on the decoding of Reed-Solomon (RS) codes has yet to be explored. In this paper, we propose a RS decoder based on deep learning for the first time, and pave a new way to improve the existing RS decoding algorithms. We exploit a deep neural network (DNN) to estimate the error numbers of the received codewords, and according to the estimation results, a novel decoder is designed, which can adjust the most suitable decoding method to each received codeword automatically. Experiments show that for (7, 3), (15, 9) and (63, 55) RS codes, the average computational complexity of our decoder can be reduced by 68.96 %, 62.38 %, 50.61 % respectively compared with the HDD-LCC algorithm.**

***Index Terms*—Deep learning, Reed-Solomon (RS) codes, deep neural network (DNN), HDD-LCC, classification decoding.**

I. Introduction

As one of the most frequently used error control codes, Reed-Solomon (RS) codes have been widely employed in digital communication and data storage systems. In general, hard decision decoding (HDD) algorithm and algebraic soft decision decoding (ASD) algorithm are main decoding methods for RS codes. Compared with HDD, ASD algorithm provides a potential for decoding errors beyond $d_{min}/2$, where d_{min} is the minimum distance of the code. And the better error performance obtained by ASD, the higher computational complexity is required. However, the existing algorithms employ the same decoding process for all received codewords and have poor adaptability, resulting in unnecessary decoding complexity. If we could select different decoding methods to different received codewords, the effect is definitely better than any single decoding algorithm. However, it is difficult to distinguish different received codewords before decoding and there are rare related methods.

Recently, deep learning has been applied in many areas and provided remarkable performance improvements. Meanwhile, considerable attention has been paid to deep learning based channel decoding. In [1], Gruber *et al*. decoded noisy codewords utilizing neural network (NN) and achieved near maximum likelihood performance, yet the method only works for rather small code lengths. Besides, Bennatan *et al*. [2] claimed that NN can estimate the channel noise when the reliabilities and syndrome are fed. The approach reduces the reliance on simulate codewords during training thus overcomes the overfitting problem. In [3]–[6], multiplicative weights were assigned to the edges of the belief propagation (BP) Tanner graph and trained by NN to obtain a better performance. What's more, an iterative belief propagation-convolutional neural network (BP-CNN) architecture was proposed in [7] to deal with the correlated noise existing in practical communication systems. These studies show that deep learning performs well in channel decoding and provides a new perspective for decoding optimization. However, almost all the related studies are based on binary codes (polar codes, LDPC codes and BCH codes), while few research have explored the beneficial effect of deep learning on nonbinary codes.

In this paper, we apply deep learning to the decoding process of nonbinary RS codes for the first time to estimate the error numbers of received codewords. According to the estimation results, a classification decoding algorithm is designed to distinguish different received codewords and select the best decoding methods later. Thus the proposed novel decoder can automatically adjust the most suitable decoding method to each received codeword and maintain high decoding efficiency. The proposed deep neural network and decoder are tested for (7, 3), (15, 9), (63, 55) RS codes and the results illustrate the functionality and effectiveness of our method. On the condition of similar decoding performance, the average computational complexity of our decoder is more than 50% lower than that of the HDD-LCC algorithm.

II. Background

A. deep neural network

Deep neural network (DNN), also known as multilayer perceptron (MLP) or deep feedforward neural network. As shown in Fig. 1, a DNN has multiple hidden layers between input layer and output layer. Each layer of DNN contains many connected neurons. Each neuron takes a weighted sum of all inputs, adds a constant called bias, and then propagates the result through a nonlinear activation function.

Assuming that the input and output of DNN are x and y, the DNN can be abstracted as a mapping function f :

$$y = f(x; \theta), \tag{1}$$

where θ denotes the weights and biases of the neurons.

By training the parameter θ, the best function approximation of the mapping function can be obtained. In order to find the

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

optimal parameter set, back propagation algorithm [8] with gradient descent optimization method [9] is commonly used.

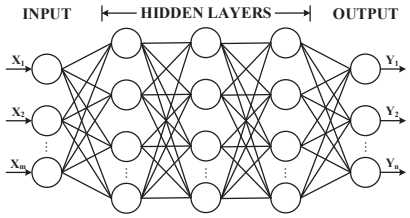

Fig. 1. The structure of DNN.

B. decoding of RS codes

(n, k) RS codes are nonbinary codes constructed over Galois field $GF(2^m)$, where n ($n = 2^m - 1$) is the code length and k is the number of information symbols. Each symbol of RS codes can be represented by m bits. The d_{min} of the RS code is $2t+1$ ($t = (n-k)/2$), and the code is capable of successfully decoding t or fewer symbol errors.

The decoding process of HDD algorithm is presented in Fig. 2. The received codeword polynomial $r(x)$ is first used to calculate the syndrome $S(x)$ with $2t$ components. The syndrome is then sent to the KES module, which evaluates the error polynomial $\Lambda(x)$ and the error value polynomial $\Omega(x)$. Next, we get the error polynomial $e(x)$ by Chien search and Forney algorithm. Finally, the transmitted codeword polynomial $c(x)$ is obtained by calculating the sum of $r(x)$ and $e(x)$.

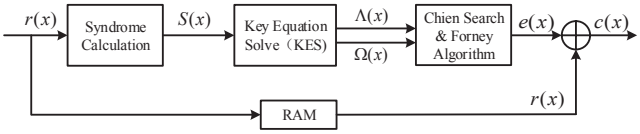

Fig. 2. HDD algorithm.

ASD algorithms for RS codes mainly include the Koetter-Vardy (KV) [10], bit-level generalized minimum distance (BG-MD) [11], and low-complexity Chase (LCC) [12] algorithms. Compared with other ASD algorithms, the LCC algorithm, which tests 2^η vectors with maximum multiplicity one, can achieve the same performance with lower complexity. To further reduce the complexity of LCC, the HDD-LCC algorithm [13], [14] was proposed (see Fig. 3), which leads to saved area and shorter latency without any decoding performance penalty.

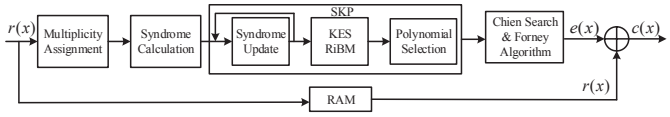

Fig. 3. HDD-LCC algorithm.

HDD-LCC algorithm uses multiplicity assignment module to generate 2^η test vectors, and decode them by the HDD algorithm successively. Then, the polynomial selection module selects the best decoding result. Applying the HDD to each test vector endow the HDD-LCC decoder with potential for handling more than t errors.

III. PROPOSED REED-SOLOMON DECODER BASED ON DEEP LEARNING

A. RS decoding analysis

Suppose that (n, k) RS codes are employed for error control in communication system. And there are three types of received codewords based on the number of symbol errors as follows:

- Type I : the received codeword contains no error;
- Type II : the received codeword contains t or fewer errors;
- Type III: the received codeword contains more than t errors.

Let P_c, P_d, P_e denote the probabilities of these types, respectively. These probabilities depend on the channel error statistics and add to 1 (i.e., P_c + P_d + P_e = 1).

According to the channel decoding theory, if there is no error in the current received codeword (type-I error), no correction is necessary. If t or fewer errors has occurred (type-II error), the error pattern can be successfully decoded by the HDD algorithm. And the ASD algorithm with high complexity is only necessary for the type-III errors. Taking (63, 55) RS codes for example, we test these probabilities over an Additive Gaussian Noise (AWGN) channel and binary phase shift keying (BPSK) mapping, and the results are presented in Table I.

TABLE I
THE PROBABILITIES OF THREE TYPES FOR (63, 55) RS CODES

Pr	SNRs						
	5.0	5.4	5.8	6.2	6.6	7.0	avg
P_c	0.0279	0.0725	0.1526	0.2675	0.4104	0.5563	0.2479
P_d	0.7061	0.8127	0.8083	0.7223	0.5875	0.4433	0.6800
P_e	0.2660	0.1149	0.0391	0.0102	0.0021	0.0004	0.0721

Table I shows that for signal-to-noise ratios (SNRs) ranging from 5dB to 7dB, the average share of P_e is less than that of P_c and P_d. In addition, as the SNR increases, P_e decreases monotonously, and even negligible for high SNRs. These probabilities indicate that it is really unnecessary to apply the ASD algorithm to all received codewords. If we could get the types of received codewords before decoding and select proper decoding methods later, the average complexity of decoding will greatly decrease.

However, it is difficult to distinguish these types before decoding. Especially for type-II and type-III errors, the type of each received codeword is not necessarily related to the syndrome or the channel output, which is difficult to find the relationship and regularity. Therefore, we cannot directly obtain these types simply by calculating syndromes or measuring channel information. We figure out that deep learning can extract the features of the received codewords and tackle this problem well.

B. system design

In this section, we introduce the decoding system based on deep learning in details. Our decoder framework is illustrated in Fig. 4.

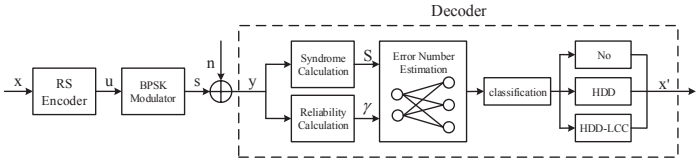

Fig. 4. Decoder framework.

At the transmitter, a message vector **x** of K ($K = m \times k$) information bits is encoded with the (n, k) RS encoding scheme to generate a N ($N = m \times n$) bit codeword vector **u**. The codeword **u** is then mapped to a symbol vector **s** through the BPSK modulation. The symbol vector **s** is later transmitted over the AWGN channel. At the receiver, the received vector **y** can be written as :

$$y = s + n, \tag{2}$$

where n $\sim N(0, \sigma^2)$ represents a white Gaussian noise.

In our system, we use a DNN to estimate the error numbers of the received codewords. Inspired by paper [2], we calculate the syndrome and reliabilities from the received vector y, and feed them into DNN.

In order to calculate the syndrome, we first make a hard decision on the binary received vector y, then convert it into a nonbinary received codeword r with length n. And the 2t syndrome components S_1, S_2, …, S_{2t} can be computed as follows:

$$S_i = r(\alpha^i) = \sum_{i=0}^{n-1} r_j(\alpha^i)^j, 1 \le i \le 2t, \tag{3}$$

where α is the primitive element for $GF(2^m)$.

The conventional method for calculating the reliabilities of RS codes is complicated and comprehensively described in [12].To simplify the calculation process, we use the idea of RCMA algorithm [15], which utilizes bit-level received voltage y to get each symbol reliability values:

$$\gamma_i = \min_{0 \le k \le q-1} \{|y_{i,k}|\}, 0 \le i \le n-1 \tag{4}$$

The construction of our DNN includes a training phase and a testing phase, respectively. In the training phase, we get the best functional approximation by minimizing the loss function over given syndrome and reliabilities. And the testing phase of the DNN is used in the decoding process to estimate the error numbers by the obtained functional approximation. Some details about the DNN will be exhibited in Section IV.

According the error numbers estimated by DNN, we propose a classification decoding algorithm based on thresholds, which is listed in Algorithm A. Considering the good performance and lower complexity of the HDD-LCC algorithm, we use it to implement the classification decoding.

Th1 and Th2 are thresholds determined by the learning effects of the DNN, and they greatly affect performance and complexity of the proposed RS decoder. To reduce the decoding complexity with a minor performance penalty, the thresholds we select are the maximum values which can successfully classify almost all type-III errors and ensure that type-II errors are not misclassified as type-I errors. Our selection of the thresholds shows great decoding performance at the expense of reduced classification accuracy and increased partial decoding complexity. Certainly, we can also obtain a complexity-performance tradeoff by selecting the thresholds reasonably.

Algorithm A: The classification decoding algorithm

Input: DNN estimation output num_error
Output: decoded codeword x'
if num_error $<$ Th1, then /*type-I error */
  output the received codeword r
else if Th1 $\le$ num_error $<$ Th2, then /*type-II error */
  output the decoding result of HDD
else /*type-III error */
  output the decoding result of HDD-LCC
endif

It should be noted that the proposed decoder and classification decoding algorithm are suitable for RS codes of arbitrary length, but the effects are limited by the DNN learning ability which is closely related to the DNN structure and hardware platform.

IV. EXPERIMENTS

A. details about the DNN

Our DNN is implemented on the Tensorflow framework [16] and an NVIDIA GTX 1070 Ti GPU is used for accelerated training. The structure of our DNN is 64-32-16, which employs three hidden layers with 64, 32, and 16 nodes.

To train the DNN, we simulate the transmission of the all zero codewords over AWGN channel and BPSK modulation, and the training SNR is selected as 6 dB. We generate 10^6 codewords in total and the mini-batch size is 120. In the testing phase, we randomly generate 10^5 testing codewords. We employ the mean squared error (MSE) as the loss function:

$$L_{MSE} = (E - \hat{E})^2, \tag{5}$$

where E and $\hat{E}$ are the actual error numbers and the DNN estimation error numbers, respectively. Since $\hat{E}$ is a nonbinary value, we obtain $\hat{E}$ by incurring a RELU function at the end of DNN.

To initialize and train the DNN, the Xavier initialization method [17] and Adam optimization method [18] are used. The training epoch and learning rate are set to 10^6 and 0.005.

B. performance of the DNN

To assess the performance of DNN, two performance metrics are measured in our experiment. The first one is the error rate of error number estimation, which measures the performance of DNN by comparing the DNN actual outputs and expected outputs. Another one is the error rate of error type estimation, which is closely related to the performance of classification. And we get the error type estimation by rounding the DNN output and comparing it with 0 and t respectively.

The results of two metrics are exhibited in Fig. 5. The solid lines in the figure are the error rate of error number estimation

on (7, 3), (15, 9), (63, 55) RS codes. It can be seen that DNN estimation achieves considerable accuracy for very short length (ie. (7, 3) RS codes). And with the increase of the code length, the estimation result of the same DNN gradually becomes worse. The dotted lines are the error rate of error type estimation, which achieve acceptable results at each code length.

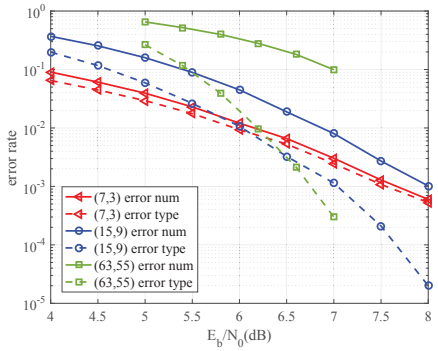

Fig. 5. The error rate of DNN estimation.

C. selection of the thresholds

As mentioned before, two thresholds are defined in the classification decoding algorithm. And in this section, we display how to select the thresholds combined with our experiment results. According to our test results, when Th1 is 0.5, all the type-I errors can be distinguished without any mistakes, thus Th1 is selected as 0.5. And the selection of Th2 is more complicated. We expect that Th2 is the maximum value which could classify almost all type-III errors successfully. To select the optimal Th2, we test the change of misclassification probability of the type-III errors with Th2 (see Fig. 6). With the increase of Th2, the misclassification probability increases constantly. When the code is a bit longer, the probability may change rapidly, due to that DNN trained many error numbers to a same number. According to the curves in the Fig. 6, for (7, 3), (15, 9) and (63, 55) RS codes, we select 1.8, 2.0 and 2.3 as the value of Th2.

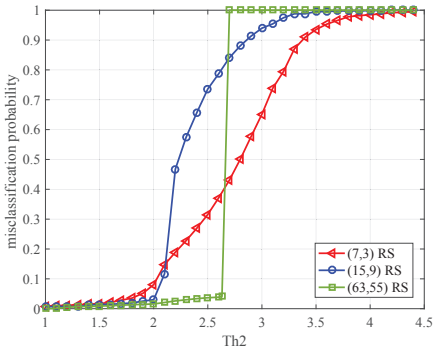

Fig. 6. Misclassification probability of type-III errors.

D. performance of the decoder

In this section, we compare the decoding performance and complexity of our proposed decoder with other algorithms, and the results are exhibited in Fig. 7.

The comparison of the frame error rate (FER) are shown in Fig. 7(a)(c)(e). Obviously, our proposed decoder based on the selected thresholds can achieve similar decoding performance with HDD-LCC algorithm. In addition, the decoding time (10^5 frames) of different decoding algorithms are compared under the same conditions to evaluate the computational complexity of the algorithm. As presented in Fig. 7(b)(d)(f), our proposed decoder with classification decoding algorithm uses less decoding time than the HDD-LCC algorithm in all range of SNRs. And as the SNR increases, the computational complexity of the proposed decoder decreases, even lower than that of HDD algorithm. Compared with the HDD-LCC algorithm, for (7, 3), (15, 9), (63, 55) RS codes, the average computational complexity of the decoder is reduced by 68.96 %, 62.38 %, 50.61 % respectively, which verifies the high-efficiency of our decoder.

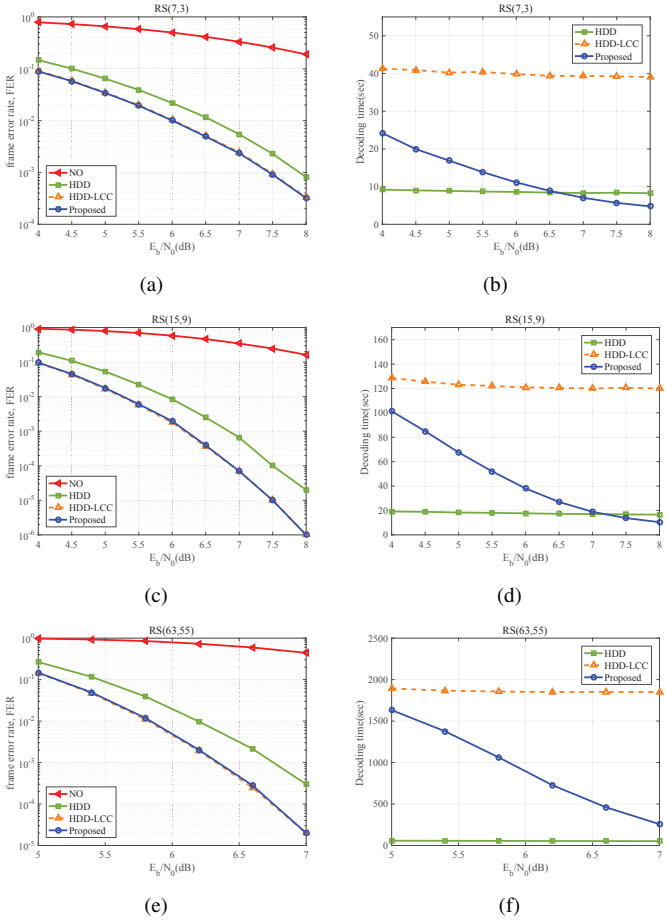

Fig. 7. Comparison of the decoding performance and complexity with code (a)(b) RS (7, 3) (c)(d) RS (15, 9) (e)(f) RS (63, 55)

V. CONCLUSION

In this paper, a novel RS decoder based on deep learning is proposed, which could adjust the decoding methods to the error numbers of the received codewords. And our proposed decoder is proved to be more efficient compared with the HDD-LCC algorithm. Future work will focus on the hardware performance of our decoder.

REFERENCES

[1] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," *in Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, 2017, pp. 1–6.

[2] A. Bennatan, Y. Choukroun and P. Kisilev, "Deep Learning for Decoding of Linear Codes – A Syndrome-Based Approach," *in Proc. IEEE Int. Symp. Inf. Theory(ISIT)*, 2018, pp. 1595–1599.

[3] E. Nachmani, Y. Beery, and D. Burshtein, "Learning to decode linear codes using deep learning," *in Proc. Annu. Allerton Conf. on Commun.,Control, Comput.*, 2016, pp. 341–346.

[4] E. Nachmani, E. Marciano, D. Burshtein, and Y. Beery, "RNN decoding of linear block codes," *arXiv preprint arXiv:1702.07560*, 2017.

[5] E. Nachmani, Y. Bachar, E. Marciano, D. Burshtein and Y. Beery "Near Maximum Likelihood Decoding with Deep Learning," *in Proc. Int. Zurich Seminar on Inf. and Comm.*, 2018.

[6] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein and Y. Beery, "Deep Learning Methods for Improved Decoding of Linear Codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[7] F. Liang, C. Shen and F. Wu, "An Iterative BP-CNN Architecture for Channel Decoding," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 144–159, Feb. 2018.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] R. Kotter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inf. Theory.*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[11] J. Jiang and K. R. Narayanan, "Algebraic Soft-Decision Decoding of ReedCSolomon Codes Using Bit-Level Soft Information," *IEEE Trans. Inf. Theory.*, vol. 54, no. 9, pp. 3907–3928, Sept. 2008.

[12] J. Bellorado and A. Kavcic, " low-complexity method for Chase-type decoding of Reed-Solomon codes," *in Proc. of IEEE Intl. Symp. on Info. Theory*, 2006, pp. 2037–2041.

[13] F. Garcia-Herrero, J. Valls and P.K. MeherHigh, "High Speed RS(255, 239) Decoder Based on LCC Decoding," *Circuits Syst. Signal Process.*, vol. 30, no. 6, pp. 1643–1669. Jun. 2011.

[14] W. Zhang, H. Wang and B. Pan, "Reduced-Complexity LCC ReedC-Solomon Decoder Based on Unified Syndrome Computation," *IEEE Trans. VLSI. Systems*, vol. 21, no. 5, pp. 974–978, May 2013.

[15] X. Peng, W. Zhang, W. Ji, Z. Liang and Y. Liu, "Reduced-Complexity Multiplicity Assignment Algorithm and Architecture for Low-Complexity Chase Decoder of Reed-Solomon Codes," *IEEE Commun. Lett.*, vol. 19, no. 11, pp. 1865–1868, Nov. 2015.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis and *et al*., "Tensorflow: A system for large-scale machine learning," *in Proc. 12th USENIX Symp. on OS Design and Implementation (OSDI)*, 2016.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." *in Proc. Int. Conf. On Artificial Intelligence and Statistics(AISTATS)*, 2010, pp. 249–256.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *in Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.