Final Report

20212245 kimheeseo

December 16, 2022

Abstract

It is a GEDU 501-02's Final Report.
Course : GEDU501-02
Assignment name : Research Report
Prof.Name : Natasha Powell

1 Introduction

With its recent development, deep learning has shown great achievements when applied to areas outside the computer science domain. Therefore, deep learning has also been applied to physical layer communication research such as channel coding and antenna technology. There are two reasons why deep learning is used in communication systems: performance improvement and complexity reduction. Based upon the first reason, deep learning technology with decoding is applied to communication systems to achieve better performance. Before the integration of deep learning, noise, which is added when a message passes through a channel, can prevent the receiver from receiving the message. To improve this, there is a process called encoding and decoding in the classical communication domain. There is a popular decoding method, known as Belief Propagation Algorithm (BPA), also named Sum Product Algorithm (SPA). This algorithm has good performance, but it is composed of many multiplication operations. Therefore, the longer the length of the message used as an input value, the more complicated calculation is. In such circumstances, the second reason to use deep learning, reducing complexity is important. To solve this complexity problem, there is a solution called min-sum algorithm (MSA). In this min-sum algorithm, the complexity problem was improved, but performance loss degradation occurred. To properly adjust this trade-off relationship between performance and complexity, there are two algorithms: 1. Normalized min-sum algorithm (NMSA) that multiplies correction factor values, which are constant values, from the check node update process and 2. Offset min-sum algorithm (OMSA) that adds or subtracts correction factor value from check node update process. By incorporating deep learning in this way into existing communication systems, the correction factor is optimized. Remarkably, results from the use of these algorithms showed better performance than MSA and improved complexity than BPA.

Recently, many researchers have been actively researching methods to incorporate deep learning into channel coding. A formative study, by Nachmani, used deep learning in the decoding process. By setting different weight values at the edges connecting check nodes (CN) and variable nodes (VN), [1] improved performance by reducing the effect of small cycles in tanner graphs during the decoding process. In [2], unlike deep learning's application to BPA, Lugosch applied it to Offset Min-Sum(OMS). Deep Learning was used to obtain OMS's optimized correction factor value. The reason why improving the complexity problem is important is that the complexity increases as the amount of computation increases. Accordingly, additional high-performance hardware is required, or heat may occur, resulting in permanent hardware problems. Therefore, to improve this, complexity is one of the problems that hardware must solve. Particularly, OMS is an algorithm consisting of addition and subtraction, not a more complex multiplication calculation, so it is a suitable method for an algorithm to apply to hardware such as semiconductors, because of its low complexity. This algorithm is called a neural offset min-sum (NOMS). [3] conducted similar research where a neural normalized min-sum (NNMS) was proposed using an optimized correction factor through deep learning. Alternatively, Wang et. al suggested a different method. In order to improve the complexity problem, they used a sharing method that uses the same correction factor value for each iteration, unlike recent researches that used different correction factor values for each iteration and node. This algorithm is called Shared Neural NMS (SNNMS).

The aforementioned research used deep learning to optimize the correction factor. However, there is a research focusing on refining the deep learning architecture for this application. Deep Learning has several architectures such as Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Recurrent Neural Networks (RNN). [4], a seminal work, used an 'RNN'. This RNN utilizes not only current inputs but also past data for learning. Remarkably, this first research to incorporate 'RNN' into the decoding process showed similar performance to prior research using fewer parameters. Subsequently, the researcher improved performance by incorporating "relaxation" into the RNN architecture in [5]. The purpose of using this relaxation concept was to determine how much of the previous data to use. Notably, this method optimized the decoder relaxation factor through deep learning as opposed to the previous method, brute force simulation. However, 'RNN' has two limitations. To be specific, input vectors are entered sequentially to enable sequential data processing, but a 'parallelization operation' is not possible. In this case, the derivative value of Hyperbolic Tangent (tanh), an activation function of RNN, is used. However, there is a disadvantage that back propagation information is rarely transferred because a vanishing gradient occurs. To solve these problems, our method used 'pruning'. It is a method of reducing the parameters of the model by removing the connection of the weights of the model that are less important. Through 'pruning', our method reduces the number of parameters, which solves the computational volume and complexity problem, but has the disadvantage of losing information. A similar method, 'drop out', is performed for the purpose of regularization. The difference is that this is applied by not using part for a while when learning, storing it in a way that it is used again later, and then using it again. Nevertheless, the proposed method was applied focusing on the advantages of improving pruning complexity. In channel coding, there are many methods to compare performance; this study used Bit Error Rate (BER), which is number of bits that have errors in the process of being transmitted to the number of bits received. In the simulation, performance was compared through simulation of BER and it will be shown that performance is better when using our proposed method than when using OMS.

2 Result

This research aims to further improve the performance of conventional NOMS by utilizing "pruning". The prior decoding method trained beta values of OMS to optimize the algorithm. Since beta value changes depend on the importance of result, beta is used as an indicator of importance. Therefore, A new method is proposed, based on the idea that removing unimportant values would not significantly affect performance and would reduce the amount of computation. The proposed method uses [1] "pruning", a method of reducing the number of model parameters. The reason for using this method is to reduce amount of computation to improve complexity. The novelty of proposed method is to present a new criterion for pruning. Beta value can have different performance depending on its changes as shown in Fig 1., so this work focused on finding the optimal beta value.

2.1 Process

Since Proposed method is based on OMS, first mention OMS in detail. Here, "N" of NOMS was named because DNN was used to find the optimal beta value of OMS. The Neural Offset min-sum (NOMS) equation is as follows.

$$\mu_{c,v}^{t} = \mathrm{ReLU}(\min_{v'}(|\mu_{v',c}^{t}| - \beta_{c,v}^{t})) \prod_{v'} \mathrm{sign}(\mu_{v',c}^{t}),$$

where $\beta_{c,v}^{t}$ is a is a learnable offset parameter for the edge connecting CN c to VN v during iteration 't'. Simulation proceeds through the following process. First, to apply neural network to decoding process, DNN's structure is made into an unrolled structure. In Fig.2, the 'odd' th layer represents VN layer, and the 'even' th layer represents CN layer. In particular, I optimized beta values assigned to the check node update and removed the check node value based on a 'special criteria': making the small value zero is method of removing the unimportant value.

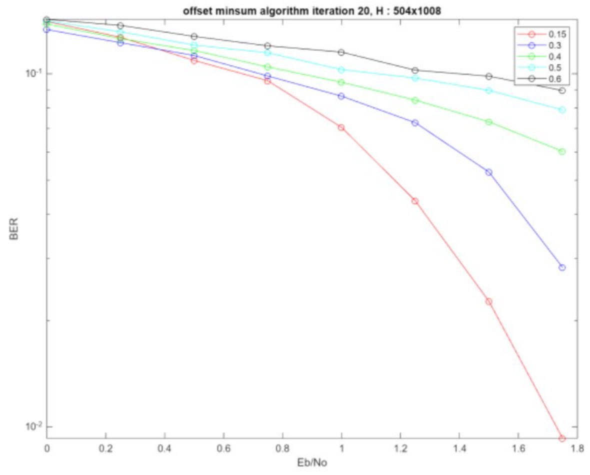

Figure 1: BER performance depending on beta value

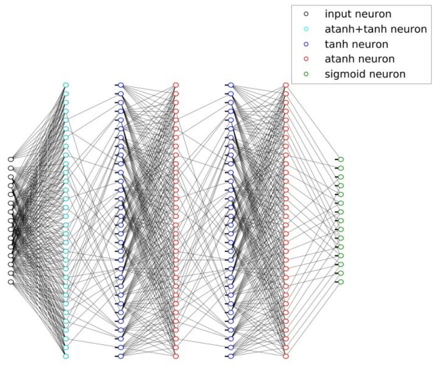

Figure 2: Deep Neural Network Architecture for BCH (15,11) with 5 hidden layers which correspond to 3 full BP iterations in [1]. It is noteworthy that this structure has been used as a basic model for decoding algorithm research with deep learning.

2.2 Setting

A simulation was conducted using [2] 'TensorFlow', using (200, 100), (120,60) LDPC code, AWGN channel and BPSK modulation, and all zero codewords. Subsequently, noisy codewords were injected after being transmitted through an AWGN channel. Then, Bit Error Rate (BER) was measured in the decoded codeword at the network output. Training was conducted using Batch Gradient descent and [3] Adam optimizer was used with a learning rate equal to 0.0001.

2.3 Pruning

Remarkably, the simulation shows similar performance to traditional research methods (OMS) while having a reduced complexity compared to these traditional methods.

Proposed method sets Optimal beta value to '0.15' and Target value. And input value is randomly set to trainable beta value and train to reduce difference from target value. Fig 3. shows that initialization value of beta value before training. Then, the error value is found through difference between the estimated and target value, and the optimal value is found through the gradient descent value.

$\beta = \beta - \eta \frac{\partial error}{\partial \beta}$

where η is learning rate (0.0001), β is trainable beta value and error is difference between target value and estimated value obtained through training. During training process, I judged that the value of beta in last iteration is greater than or less than 15 % of the target value and pruned it. As Fig 4.

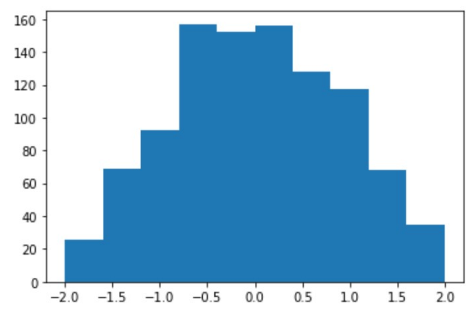
Figure 3:   Distribution of beta values before training(x-axis : beta value, y-axis : number of units)

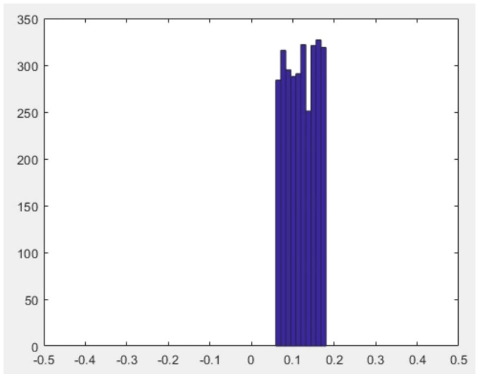
Figure 4:   Distribution of beta values after training(x-axis : beta value, y-axis : number of units)

shows, the “number of beta” is significantly reduced and the performance is evaluated with Bit Error Rate (BER) similar to the traditional method (OMS) in Fig 3. Therefore, by reducing the number of beta and reducing amount of computation, performance is shown to be superior through the fact that performance is similar to traditional methods while having a improved complexity to traditional methods. At this point, values greater than or less than 15% of 0.15 are considered insignificant in the result value, and zeroed out. This reduces the amount of computation by reducing the values not needed. In Fig 5., it can be seen that the proposed method has similar performance while improving complexity by comparing BER of proposed method and OMS.

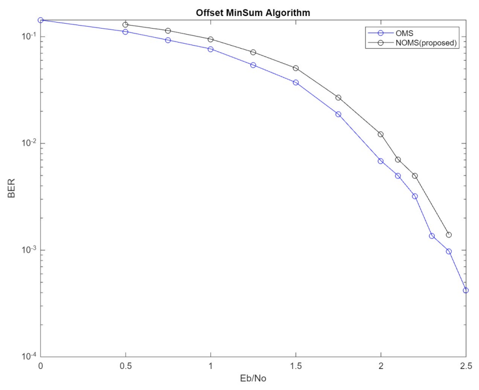

Figure 5:   BER performance comparision between OMS and NOMS)

Table 1 shows the number of beta, and 3024 mean when 0.15 is set at all 1 positions of the parity check matrix (H). Although the beta was reduced by more than half in all dB, it was confirmed through Fig. 5 that the performance was similar.

	0dB	0.5dB	0.75dB	1dB	1.25dB	1.5dB	1.75dB	2dB	2.1dB
OMS	3024	3024	3024	3024	3024	3024	3024	3024	3024
NOMS	1490	1484	1462	1479	1449	1442	1493	1464	1536

Table 1: Comparison of Beta numbers of OMS and proposed methods according to dB

2.4 iteration and various codes

Since performance and complexity have a trade-off relationship, if the number of iterations or size(length) of code are increased to improve performance, performance improves, but complexity also increases. In this section, while a simulation is performed by changing 'iteration' and 'code size', it is confirmed whether condition is also true in the proposed method.

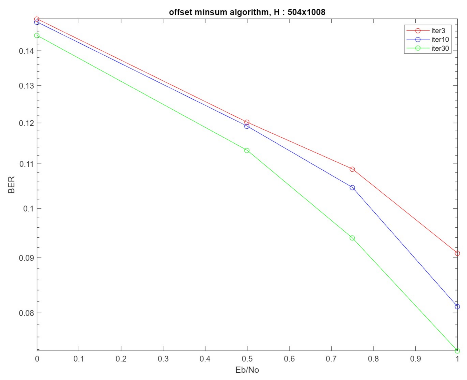

Figure 6: Results depending on iteration)

Fig 6. is a comparison of BER performance that varies for different iterations. It confirms that performance improves as the iteration increases.

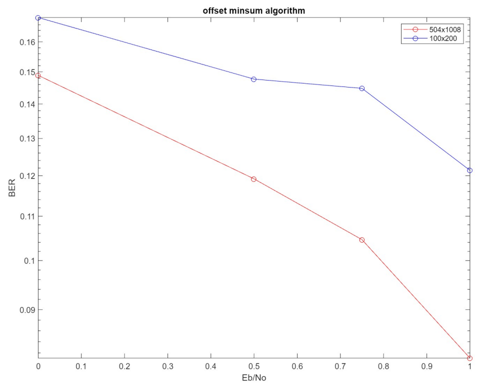

Figure 7: Results depending on code length)

Fig 7. compares BER performance of those with different code lengths, and also confirms that performance improves as code length increases. Through two simulation result(fig 6, 7), it was confirmed that the existing law"As Code length and the number of iterations increase, decoding performance improves." was established. But, complexity increases. Therefore, proposed method improved its point.

3 Conclusion

Recent research [1] has shown decoding performance improvement with the recent incorporation of deep learning into Channel Coding, which has led to the emergence of methods such as [4] neural offset minsum (NOMS), [5] neural normalized misum (NNMS), and shared neural normalized minsum (SNNMS). Studies that proposed 'NOMS' and 'NNMS' mainly focused on performance improvement, whereas 'SNNMS' research focused on improving complexity. In particular, since there is a trade-off relationship between complexity and performance improvement, an appropriate value should be adjusted, and the proposed method focused on improving complexity. At this time, complexity was improved by reducing the amount of calculation through a method called 'pruning'. This method reduces the amount of computation by training different beta values for each edge between the variable node (VN) and check node (CN) in OMS and removing small values per each row. The reason is that the size of value represents the importance of decoding as indicator. Through simulation, it is shown that performance of proposed method shows similar performance to prior one (OMS) while improving complexity. Through this, it is easier to apply hardware such as semiconductor, communication equipment and so on. For that reason, "samsung electronics" and "huawei" are currently focusing on it. Future research can apply the proposed method in various code as well as in BCH, Polar code, and determine whether the proposed method can be applied to NNMS and SPA.

4 Reference

[1] Nachmani, Eliya, Yair Be'ery, and David Burshtein. "Learning to decode linear codes using deep learning." 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2016.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," Software available from tensorflow. org, vol. 1, 2015.

[3] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[4] Lugosch, Loren, and Warren J. Gross. "Neural offset min-sum decoding." 2017 IEEE International Symposium on Information Theory (ISIT). IEEE, 2017.

[5] Wang, Qing, et al. "A model-driven deep learning method for normalized min-sum LDPC decoding." 2020 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2020.