EECE695D-01's class final report

kim heeseo 20212245

Abstract

While taking "EECE695D-01 class", I studied various artificial intelligence models. Report investigated how algorithms learned in class are used in the research field, and present future research directions.

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 first reason, deep learning technology with decoding is applied to communication systems to achieve better performance. Before integration of deep learning, noise, which is added when a message passes through a channel, can prevent the receiver from receiving message. To improve this, there is a process called encoding and decoding in 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, Second reason to use deep learning, reducing complexity is important. To solve complexity, there is a solution called min-sum algorithm (MSA). In MSA, Complexity problem was improved, but performance loss degradation occurred. To properly adjust trade-off relationship between performance and complexity, there are two algorithms: 1. Normalized min-sum algorithm (NMSA) that multiplies correction factor, which are constant, from the check node (CN) update process and 2. Offset min-sum algorithm (OMSA) that adds or subtracts correction factor from CN update process. By incorporating deep learning in this way, Correction factor is optimized. Remarkably, results from 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 decoding process. By setting different weight w values at edges connecting CN and variable node (VN), [1] improved performance by reducing effect of small cycles in tanner graphs during the decoding process. It is called Weighted Belief Propagation decoding Algorithm (WBP). In [2], unlike deep learning's application to BPA, Lugosch applied it to OMSA. Deep Learning was used to obtain OMSA's optimized correction factor. The reason why improving the complexity problem is important is that complexity increases as the amount of computation increases. Accordingly, additional high-performance hardware is required, or resulting in permanent hardware problems. Therefore, to improve this, complexity is one of problems that hardware must solve. Particularly, OMSA is an algorithm consisting of addition and subtraction, not a more complex calculation, so it is a suitable method for an algorithm to apply to hardware such as semi-conductor. 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 same correction factor for each iteration. This algorithm is called Shared Neural NMS.

Channel Coding Research trend using deep learning up to 2020 focused on optimizing Correction factor of conventional algorithms. However, recently, researchers have been researching performance and complexity improvement by applying pruning, transformer, knowledge distillation (KD) and so on. These researches were investigated in report, and a new research direction was presented.

2. method

2.1. Pruning

In [4], **Pruning** is applied to tailor an overcomplete parity-check matrix (PCM),whose rows have linear dependency, to BP decoding using machine learning. At this time,Pruning and Dropout are the same as those used in "regulation" and reduce parameters. However, unlike dropout, pruning is a method of reducing the numbers of w completely, resulting in loss of information but improving complexity. While WBP improves upon conventional BP decoding, its performance is still limited by the underlying PCM. So, Buchberger introduced a pruning-based approach to selecting the best PCM for each iteration of the BP decoder for short linear block codes. Training process is as follows.

1. Set overcomplete PCM by total iterations, and set **W**

(weight matrix) initialized to 1.

2.**W** are optimized using Adam optimizer.

3. After optimization has converged, find index and iteration of the lowest CN *w* and set it to zero (pruning).

4. Rerun and Training until either reach a desired number of PCM or until the loss starts diverging.

Specifically mentioned step by step, it is as follows.
In 1, Because PCM and **W** are set as many as the number of iterations, they were not tied over iterations. So, final decoder uses a different PCM in each iteration.
In 2, all messages at a single CN are weighted by the same *w* as in (1), because of reducing complexity.

$$\lambda_v^{(l)} = w_v^{(l)} \lambda_{ch,v} + w_{v\to c}^{(l)} \sum_{\tilde{c}\in N(v)} \lambda_{\tilde{c}\to v}^{(l)} \tag{1}$$

In 3, At this time, Reason for considering magnitude of **W** is that it can be interpreted as a measure of how much CN contributes to decoding. So, a magnitude of zero indicates that CN is irrelevant to decoding process.

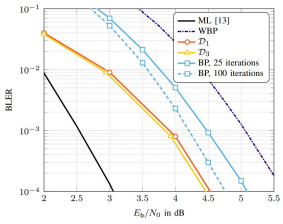

Figure 1. BLER results for the LDPC code.

Simulation Setting is CCSDS LDPC code of length 128 and rate 0.5 and CN degree of 8 with half the VNs having degree 3 and half having degree 5. In simulation result, *D1* denoted method by which PCM and **W** are optimized in proposed method, *D2* denoted method by which only PCM is optimized, **W** is set to 1, *D3* is the method by which only PCM is optimized, and **W** is untied optimized *w* over all iteration. Although *D1* and *D3* performed 6 iterations, they showed that they performed better than BP by 0.6 dB, improving complexity and improving performance with less iterations. And researcher conducted **simliar study** in [5], and a two-stage decimation process was constructed to improve the performance of NBP. Specifically, In first step, it guess the least reliable bit and iterate between connvetional NBP(CNBP) decoders. And make a list accordingly. In Second step, iterates between a CNBP decoder and learned decimation. At this time, a neural network (NN) is used to determine which a posteriori LLR to decimate.

2.2. Transformer

In [6], Unlike conventional methods, it first proposed a **Transformer** architecture-based model free decoder, which was mainly used in natural language processing such as machine translation, to soft decoding of linear codes with arbitrary block length, and applied interaction between Algorithm code and Bits model through the adaptive self-attention (SA) module.

For **transformers**, Input sequence is first embedded into a high-dimensional space that combines positive embeddings for each element, and its value is propagated through a number of normalized SA and feed-forward blocks. In the case of SA, first, three vectors, **key**, **value**, and **query**, are created from input vectors, and then association between query and key is obtained through dot product. And Divide by the square root of the key vector size to obtain a stable gradient. Then, take the softmax score and multiply it by 'value' to remove the non-critical value. Finally, add all of those values to obtain output value of SA. Additionally, if **H self-attention** functions are applied to the input, it becomes multi-head self-attention (MH-SA). Proposed method is changed as follows for "error correcting code". First, Do high dimensional embedding of n-dimensional of the channel output(y) $(\phi_i)_{i=1}^{2n-k}$.

$$(\phi_i) = |y_i|W_i, \quad if \quad i \le n$$
$$(1 - 2(s(y))_{i-n+1})W_i, \quad \text{otherwise}$$

$(W_j \in R^d)_{j=1}^{2n-k}$ is denoted one-hot encoding, defined according to bit position. Depending on *i*, it is expressed by (ϕ_i) because the value of $\tilde{y} = h(y) = [|y|, s(y)]$ is composed of channel output reliability : absolute value and syndrome value. In particular, the reason why it is expressed as $(1 - 2(s(y))_{i-n_1})$ is because of BPSK process. In other words, we can see that process is bit-wise embedding through subscripts representing the position, and that it is scaled according to the value of (ϕ_i), so we can see that it is 'scaled bit-wise embedding'. The researcher referred to this process as 'positional reliability encoding'. And 'SA' for error correction code was applied, and functions and algorithms are expressed as follows.

$$g(H) : (0,1)^{(n-k)\times n} \to \{-\infty, 0\}^{2n-k\times 2n-k}$$

$$A_H(Q,K,V) = \text{Softmax}(\frac{Qk^T + g(H)}{\sqrt{d}})V \tag{2}$$

In fig 2., Researcher proposed adaptive-mask SA mechanism, which proposed a symmetric mask containing information about relationship between all bit pairs, and unmasked the position of 1 in row. Its novelty is a Transformer architecture for the decoding of algebraic block codes. It allows effective representation of interactions based on high dimensional embedding of channel output and a code-dependent masking scheme of SA module. Since it is the first proposed method, it will be necessary to demonstrate superiority of proposed method's performance under various conditions, such as various code, quantization, and Irregular PCM. And Pre-processing pruning to reduce the number of masks will be direction of future research.

Algorithm 1: Mask construction Pseudo Code

```
function g(H)
    l,n = H.shape
    k = n-l
    mask = eye(2n-k)
    for i in range(0,n-k) do
        idx = where(H[i]==1)
        for j in idx do
            mask[n+i,j] = mask[j,n+i] = 1
            for k in idx do
                mask[j,k] = mask[k,j] = 1
    return −∞(¬ mask)
```

Figure 2. Mask construction Pseudo Code

2.3. Knowledge Distillation

Prior research have tried to improve results by changing [7]the neural architecture, the loss function at the output layer or [4] finding better sparse parity check matrix. In [8], it proposed to use two novel loss terms that optimize decoder nodes. 1) sparse node loss, 2) knowledge distillation loss.

$$L = L_{ce} + \alpha L_{kd} + \gamma L_s \tag{3}$$

, where optimal α, γ is 1 and 0.01.

First term $L_{ce} = -\frac{1}{n}\sum_v B_v \log(\sigma(s_v)) + (1 - B_v)\log(1 - \sigma(s_v))$ is cross-entropy (CE), a commonly used loss function, where $B_v = (1 - X_v)/2$ is bit corresponding to transmitted symbol X_v.

Second term $L_{kd}(t) = \sum_{v=1}^{n}\sum_{c' \in N(v)} ||\sigma(\mu_{c',v}^{teacher}(t + t_o)) - \sigma(\mu_{c',v}^{student}(t))||_p^p$ is proposed KD method generated a vector of LLR values with same encoded bits vector for teacher and student networks in training, where $\mu_{c,v}^{student}(t)$ is student network messages at iteration t tried to mimic teacher network messages $\mu_{c',v}^{teacher}(t + t_o)$ at iteration $t + t_0$, messages t_0 is a look ahead parameter, and p is the norm order. More detail about it, Deep learning improves performance with deeper, more parameters, and computations, but efficient models should have good performance with smaller models. So, Tradeoff relationship between complexity and performance should be well adjusted. Therefore, KD method focuses on efficiency, and output of pre-trained **teacher network** is learned by **student network**, which is a small model to actually use. As a result, it is a way to increase the performance of model even though it has relatively few parameters.

Third term is proposed sparse loss term,$L_s(t) = \sum_{v=1}^{n}\sum_{c' \in N(v)} ||\sigma(\mu_{c',v}^{student}(t))||_p^p + \sum_{c=1}^{n-k}\sum_{v' \in M(c)} ||\sigma(\mu_{v',c}^{student}(t))||_p^p$, where p value in KD loss terms. As p value increases, gradient value increases, which is an important parameter for successful training. More detail about it, **Sparse activation regularization** imposes sparse constrained on activation with new loss function term. Because sparse PCMs such as LDPC leads to better decoding performance, Researcher proposed teaching NN decoder with sparse nodes. The reason why sparse PCMs perform well is that the number of short cycles that adversely affect decoding performance can be reduced by minimizing dependency relationship between each nodes.

To comment on personal research, researcher explained through Ablation study (AS), sparse node activation loss gives more improvement than KD loss. In the case of "Train only with KD loss", there is a degradation in low SNR regime, which is due to CE. However, researcher did not explain this by comparing results conducted only with these factors, so I judged that logic was insufficient.

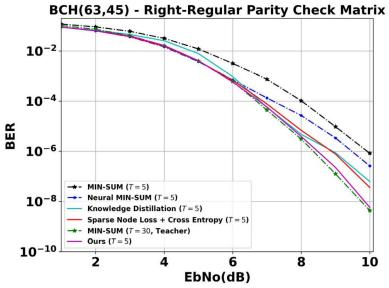

Figure 3. BER for various values of SNR for BCH codes.

3. Conclusion

Unlike prior researches that applied deep learning to optimize correction factor, performance was improved by methods such as pruning, KD, and transformer. They are new methods that have not been used in channel coding, and there are many fields that have not yet been applied. So, I am conducting a research to apply pruning to OMS, not just based on weight value, but sign value in CN node update and most papers focus on regular LDPC code, I wonder what it would be like to be irregular. One more thing, In [8], By setting γ to a very small value of 0.01, sparse node activation loss term was included in total loss term. Therefore, I thought researcher was forced to put ideas in to appeal. . In addition, since Comparison was set incorrectly in AS, I think it is also a topic of future research to understand how it affects each other's relationship and decoding performance.

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. IEEE, 2016.

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

[3] Wang, Qing, et al. "Normalized Min-Sum Neural Network for LDPC Decoding." IEEE Transactions on Cognitive Communications and Networking (2022).

[4] Buchberger, Andreas, et al. "Pruning neural belief propagation decoders." 2020 IEEE International Symposium on Information Theory (ISIT). IEEE, 2020.

[5] Buchberger, Andreas, et al. "Learned decimation for neural belief propagation decoders." ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021.

[6] Choukroun, Yoni, and Lior Wolf. "Error Correction Code Transformer." arXiv preprint arXiv:2203.14966 (2022).

[7] Nachmani, Eliya, et al. "RNN decoding of linear block codes." arXiv preprint arXiv:1702.07560 (2017).

[8] Nachmani, Eliya, and Yair Be'ery. "Neural Decoding With Optimization of Node Activations." IEEE Communications Letters 26.11 (2022): 2527-2531.