Comparing accuracy of sign-quantized layers with linear layer in small models

Choi Soon Ho, Soo-Yeon Yoon*

Kookmin Univ., *Kookmin Univ.

sosaror@gmail.com, 1104py@kookmin.ac.kr

작은 모델에서의 선형 레이어에 대한 부호 양자화 레이어의 모델 정확도 비교 연구

최순호, 윤수연*

국민대학교, *국민대학교

Abstract

Recent advancements in neural network quantization have introduced the potential of significantly reducing model sizes without compromising accuracy with sign-quantized layers only consisted of 1, 0, -1. Our study focused on the comparing accuracy of various layers with different designs. In the progress we found that the sign-quantized layers have enough potential with experiments on small models. Especially, results of chatGPT generated layer showed that sign-quantized layers have more potential with resulting better accuracy in most cases against simple modified bitlinear layer which learns similar with the previous studies, while showing its learning strategy is not yet fully disciplined with fluctuating accuracy and extreme loss.

I. Introduction

In a recent paper in The Era of 1-bit LLMs: All Large Language Models in 1.58Bits suggests that quantizing weights to 1, 0, -1 in fully connected layers of LLM-scale AI barely impacts accuracy and for larger models, it performs even better[1]. Since previous studies mainly focused on accuracy equivalence and memory efficiency of quantized models, we focused on accuracy and size of quantized models.

II. Problem definition

In this research, we are focusing on the accuracy of sign-quantized layers, probing the minimal model size where it remains effective. The size of the model was measured simply by the width of each layer and the number of layers consisting the model which called as depth.

III. Design of sign-quantized layers

In the Bitnet: Bit-Regularized Deep Neural Networks paper, BitNet presented an 8-bit quantization of weights and inputs[2]. Since then, several studies have continued, and in particular, the Bitnet: Scaling 1-bit transformers for Large Language Models, showed a 1-bit transformer model with comparable performance against 16-bit model [3].

In this study, we used default linear layer from PyTorch and 8-bit Bitlinear layer of BitNet[2] as a comparison group. Actual python code of later is from https://github.com/kyegomez/BitNet?tab=readme-ov-file [4]. The code was manually implemented from GitHub due to version compatibility issue.

To focus on weight quantization, we designed two sign-quantized layers which solely quantizes weights during forward process.

First layer we designed is a modification from BitNet[2] focused on sign-quantization of weights. Modification was done by removing quantization part of input from 8-bit bitlinear and adjusting sign-quantization to the weight replacing original 8-bit quantization part. We named “mybitlinear” to this layer.

Second layer we designed is a sign-quantized layer with the following prompt to ChatGPT 4 :

“내가 1bit BitLinear 코드를 간단하게 만들어보려고 해. 이 기법은 nn.Linear 이후에 weight값을 정규화한 뒤에 값이 음수면 -1로 아니면 1로 변환하는 알고리즘이야. 변환은 바로 진행되지만 학습할 때는 변환하기 이전 weight를 통해 학습한대. nn.Linear를 expand하는 새로운 BitLinear_1bit 클래스를 만들고 싶은데 어떻게 코드를 만들어야 할까?”

Which is Korean term for :

”I'm going to make a simple one-bit BitLinear code. This technique is an algorithm that normalizes the weight value after nn.Linear and converts it to -1 or 1 if the value is negative. The conversion proceeds right away, but when learning, it is learned through the weight before the conversion. I want to create a new BitLinear_1 bit class that expends nn.Linear, how can I make the code?"

The layer learns different from the conventional learning method. While forwarding process, it keeps all the informations from the linear layer but passes signed result of regularized input. When learning, it just passes back the propagated value, and its weights learn only when the absolute value of gradient is less than 1.

This layer was supposed to be discarded after refining, but the accuracy and loss of this layer was to be discussed for its irregularity. So we decided to include this layer in the study and named it as myLittleMisborn.

IV. Experiment design

We designed experiments using MNIST and CIFAR-100 datasets to compare the accuracy of each layers in problems of different complexity. The experiment was performed with various depth (number of layers used) of 1, 3, 5, 7 and width (size of each layer) of 4, 16, 64, 256, 1024. We used Adam optimizer, CrossEntropy loss, and default Pytorch learning rate to make fair comparison among different layers. Since quantization process is similar to activation, we used ReLU as an activation function for default linear.

Depending on the size of the model, 4 to 18 epochs were trained, and tested and printed its accuracy and loss for each epoch to track the learning curve.

We compared Default Pytorch nn.Linear, 8-bit Bitlinear [4] and two implementation derived from this study : myLittleMisborn and mybitlinear.

The code used and its results are open for review at https://www.kaggle.com/code/snowian/bitnet-vs-linear/settings?scriptVersionId=174387025.

V. Result

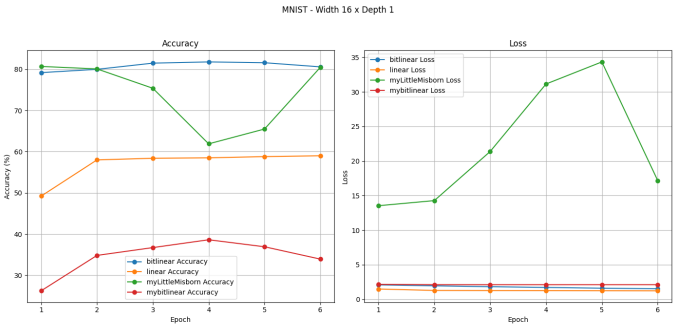

Figure 1. Accuracy and loss of width 16, depth 1 for MNIST

In figure 1, while 8-bit bitlinear results in best accuracy with stable learning curve, default linear follows with lesser result. Interesting diversity showed in this graph between mybitlinear and myLittleMisborn. While mybitlinear the “rational” layer has low accuracy, myLittleMisborn shows decent accuracy with fluctuation of both accuracy and loss.

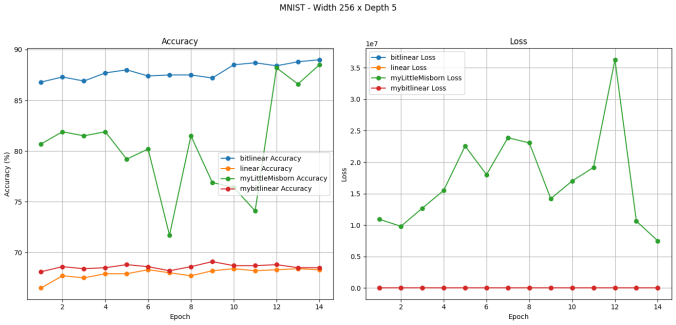

Figure 2. Accuracy and loss of width 256, depth 5 for MNIST

Figure 2 In myLittleMisborn fluctuates in both accuracy and loss greater, but it still places second with accuracy. Learning curve of myLittleMisborn seems like “avoiding low accuracy” more than converging into low loss

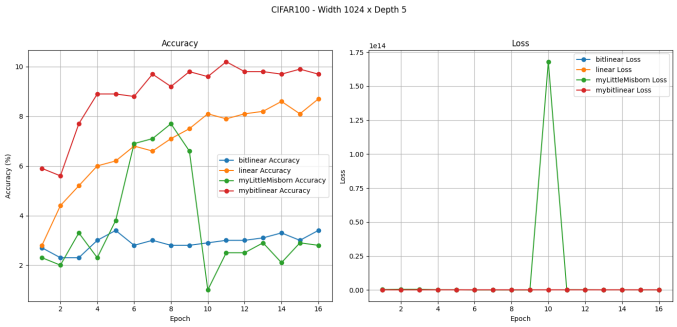

Figure 3. Accuracy and loss of width 1024, depth 5 for CIFAR-100

In Figure 3, mybitlinear results in best accuracy with stable learning curve and default linear is following. The myLittleMisborn layer fluctuates but failed to recover in 18 epochs. When it falls, the loss grows extreme even in exponential graph.

Generally, Default linear performs well as it is default. But it does not work well with width over 256 due to small dataset of this study.

8-bit Bitlinear rules most the cases with stability, but its dominance get weaker on larger models with CIFAR-100.

The modified 8-bit Bitlinear model, mybitlinear shows decent performance in the largest models.

The loss values of myLittleMisborn dominates the graph with exponential value. In the contrast, it has acceptable accuracy throughout the experiment. It also shows interesting behavior that its loss and accuracy fluctuate more as the depth goes deeper.

VI. Conclusion

Given the overall accuracy of myLittleMisborn, it is logical to assume that sign-quantization is not just a myth, and the fact that this radical layer yields higher accuracy compared to mybitlinear in most cases, leads to conclusion that current learning strategy is not very efficient for sign-quantized layers.

8-bit bitlinear shows stable and high accuracy throughout most cases. To explain this we have hypothesis that 8-bit regularization makes floats not to lose its lesser digits while forward propagation.

VII. Further research

Most ongoing models use linear layers with width more than 1024 these days. This experiment needs to be cleared with more epochs and datasets to actually be used in modern models.

To check whether the sign-quantization of the weight means more than just regularizing under 8 bits to avoid loss from underflow, we lacked experiments on 4-bit Bitlinears.

In addition, the method of increasing the range of quantization to zero already identified in Bitnet b1.58[1] also needs to be confirmed through experiments on a small scale.

Given that the variant of sign is limited, it is also possible to think about expanding this study to the complex number domain[5].

Above all, we need to discuss of different learning strategy for sign-quantized layers. Some communities study this field are discussing q-learning[6] because it is proven to converge with the set of quantized numbers.

While planning next research about straight forward estimator, which is technical base of myLittleMisborn, we found that in the Binary Neural Network[7], there was preceding study with googleNet and AlexNet models replacing the layers against MNIST and CIFAR-10, ImageNet dataset. As the result from 1.58Bits[1] case, they resulted in similar accuracy with less computational cost. However, in both studies, there was less experiment about how the width and depth of fully connected layer actually differs from the method the layer uses.

This made us to have assumption that the increase of accuracy in the study of 1.58bit[1] could have caused by the expansion of width and depths of fully connected layers part of the model as the model goes bigger. More likely because of the width in our guess.

We believe further study for bigger models with more STE layer and various structures would lead us to the optimal model for sign-quantized layers.

Also, dynamic changes to the model itself could be worth studying. To make inspiration for adding or removing of layers, we have using techinques from genetic algorithm[8] in our mind.

ACKNOWLEDGMENT

※ 이 연구는 국민대학교 학술연구비로 지원되었음(2024)

First Author : Kookmin University, Department of Computer Engineering, sosaror@gmail.com, 학생회원

°Corresponding Author : Dept of School of Software Convergence. Kook Min Univ. Seoul, South Korea., 1104py@kookmin.ac.kr, 정회원

REFERENCES

[1] MA, Shuming, et al. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits. arXiv preprint arXiv:2402.17764, 2024.

[2] RAGHAVAN, Aswin, et al. Bitnet: Bit-regularized deep neural networks. arXiv preprint arXiv:1708.04788, 2017.

[3] WANG, Hongyu, et al. Bitnet: Scaling 1-bit transformers for large language models. arXiv preprint arXiv:2310.11453, 2023.

[4] https://github.com/kyegomez/BitNet/tree/main/bitnet

[5] LEE, ChiYan; HASEGAWA, Hideyuki; GAO, Shangce. Complex-valued neural networks: A comprehensive survey. IEEE/CAA Journal of Automatica Sinica, 2022, 9.8: 1406-1426.

[6] DAYAN, Peter; WATKINS, C. J. C. H. Q-learning. *Machine learning*, 1992, 8.3: 279-292.

[7] HUBARA, Itay, et al. Binarized neural networks. Advances in neural information processing systems, 2016, 29.

[8] IBA, Hitoshi. Bagging, boosting, and bloating in genetic programming. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2. 1999. p. 1053-1060.