Master's Thesis of Cho Gyeongje

Efficient Methods for Integer only Quantization

February 2023

Graduate School of Data Science
Seoul National University
Data Science Major

Cho Gyeongje

Efficient Methods for Integer only Quantization

Name of Examiner

Submitting a master's thesis of
Data Science

March 2023

Graduate School of Data Science
Seoul National University
Data Science Major

Cho Gyeongje

Confirming the master's thesis written by
Cho Gyeongje
Month Year

Chair	____________________	(Seal)
Vice Chair	____________________	(Seal)
Examiner	____________________	(Seal)

Abstract

Transformer is the most popular Neural Network Architecture currently achieving state-of-art in various fields. Therefore, many people have tried to quantize this model so that inference can be made only with integer arithmetic (Integer-only Quantization) so that it can operate on edge devices. However, there were constraints for the general hardware and efficient calculation of non-linear functions, so the quantization method did not change significantly. Therefore, we studied new methods that can be used in Integer-only Quantization to overcome the limitations of existing methods.

Keyword : Quantization, Machine Learning, Transformer,
Student Number : 2021-24348

Table of Contents

Chapter 1. Introduction

1.1. Purpose of Research

Transformer([20]) is currently one of the most popular neural network model structures. Transformer is achieving State of Art performance in various fields such as NLP ([12], [14], [20], [2])and Computer Vision([5]). However, despite these promising results, it is becoming more difficult to actually use them due to the complex structure and the gradually increasing model size. Therefore, it is one of the essential tasks to adjust the model so that it can operate with a device which has limited resource.

Quantization([8], [7]) is one of the best ways to solve this problem which converts the parameters of the model to low precision. This process can reduce the model's accuracy but can significantly reduce the model size. In addition, by applying quantization to activation can substantially improve the inference time by changing the floating point operation required for inference to an int operation. Many previous studies have already shown that quantization can be used successfully in various model structures such as CNN([11], [22], [6]) and RNN([23]).

Many previous studies have shown that quantization can be successfully applied to Transformers as well. However, most studies([18], [4], [3]) have used the simulated quantization method, which means that even if the model parameters are stored at low precision, all or part of the values is calculated using floating points during inference. This means that it is difficult to apply to some edge devices specialized for integer arithmetic. Also, compared to Integer−only quantization, in which all calculations are performed only by Int operations, this method has difficulties in obtaining benefits in areas such as latency, power consumption, and area efficiency. In particular, this difference becomes even more pronounced when supporting fast, low-precision operations such as NVIDIA V100 and Titan RTX([10]).

To solve this problem, I-BERT([12]) and I-ViT([13]) proposed a method of compressing the transformer using integer-only quantization. They offered an approximation formula that allows non-linear functions such as GeLU([9]), Softmax, and Layer Normalization([1]) to operate only with integer and can achieve successful performance. However, as it only quantizes the model with int8, it did not show a higher compression rate compared to the method using simulated quantization.

Therefore, in this study, we propose a quantization method that can solve these limitations. We study new integer-only quantization methods using only low bits (e.g., int8, int4) and a strategy to reduce the error that occurs by using low bits. The specific details are as follows.

- We propose new quantization methods called Pseudo Float Quantization (PFQ) and Group Quantization (GQ) for Integer-only Quantization. PFQ can express a wide range of numbers using only low-bit, enabling us to effectively quantize the value of the variance skewed to 0, like the weight of the Neural Network. GQ considers the NPU(Neural Processing Unit), so that it can operate efficiently in most hardware.
- We propose an error compensation method based on Taylor Expansion to solve quantization errors in non-linear functions. As the non-linear function operates only with integers, the overflow problem can occur, but this method can reduce quantization error freely from this problem.

1.2. Related Work

Quantization refers to a method of expressing continuous values with low precision. In some cases, the model's size is reduced by quantizing the parameters of the neural network model. The inference time is greatly improved by quantizing the activation value so that the operations required for inference operate only in int. it may reduce accuracy to some extent because it cannot accurately represent the actual value. Still, many studies have confirmed that it is possible to trade off accuracy and performance at an appropriate level when using fp16 or int8.

Quantization is classified into uniform and non-uniform quantization according to the interval of quantized values. Uniform quantization is a method to keep the gap of quantized values constant, and Symmetric Uniform Quantization([11], SUQ) is the most representative. It performs quantization using only scaling and rounding, and because the operation is simple, the operation speed is fast, and it works well on all hardware. In addition, quantized values can be efficiently processed even when complex operations, such as non-linear operations, need to be applied.

Non-uniform quantization([16], [17]) is a method in which the intervals of quantized values are not constant. Since quantization is performed considering the distribution of actual values, less quantization error occurs compared to uniform quantization. Various methods, such as log and cluster, are used. Among them, the usual way mainly applied to the transformer is Binary Coding Quantization (BCQ, [19]). BCQ will express the weight parameter as the sum of the 1-bit matrix with fp32 as the weight. Since the matrix consists of 1-bit, it is possible to compress it at a high level, and it has the advantage of being able to control the quantization error by adjusting the number of matrices. However, hardware that can efficiently operate Matrix Multiplication (MMP) operation with a 1-bit matrix is required for speed improvement([18]). There is a problem that a process is necessary.

Since expressing all layers of a neural network model with low bits can significantly reduce accuracy, methods of applying different precision to each layer have been actively studied. This method is called mixed-precision quantization. Usually, quantization sensitivity is measured using the degree of quantization of each layer on the result. After that, the level of precision to be applied to each layer is determined based on this. In the case of the transformer([3], [4]), it has been confirmed through various experiments that non-linear functions such as GeLU, Softmax, and LayerNorm and the Feed Forward Layer of the Encoder/Decoder Layer are sensitive to quantization.

Chapter 2. Method

We suppose that minimizing the error caused by quantization is crucial to maintain accuracy even after performing quantization. In addition, in Integer-only Quantization, it is important that variables in each layer be quantized to share the same fp32 value in order to separate integer and float operations during inference, so we considered this condition can be satisfied.

2.1. Pseudo Float Quantization

To reduce the error caused by quantization, it is essential to express the actual value using the broadest possible range of numbers. However, there is a clear limit to the degree of numbers that low-bit can represent. We expressed the quantized value using the floating-point format to overcome this limitation. We first set an appropriate scale value and then proceeded with quantization through the same process as scaling quantization. Then, each quantized value can be expressed as two low-bit values. The actual value is expressed as follows.

$$\mathrm{r} \approx \mathrm{s} \times \mathrm{q} \approx \mathrm{s} \times 2^{q_e} \times q_m$$

q_e, q_m is expressed using b_e, b_m bit, $0 \leq \mathrm{q_e} \leq 2^{b_e} - 1, -2^{\{\mathrm{b_m}-1\}} \leq \mathrm{q}_m \leq 2^{\mathrm{b}_m-1} - 1$. When the quantized value is not accurately expressed in the form above, the quantized value can be changed to the closest value that can be expressed.

One value is expressed using an exponent of 2 because if both values are expressed as int4, three int4 values are multiplied during the actual Matrix Multiplication operation. This process deals with different precision, such as int4 and int8. This is because there is a problem with calculating. To solve this problem, an exponent of 2 is used, which replaces one multiplication operation with a bit-shift function to minimize the cost required for the process.

The scaling factor of this method becomes exponentially smaller so it cannot express large values in detail. Therefore, the quantization error of a small value can be significantly reduced, but when the value is large, the quantization error does not change. Thus, a significant effect can be seen when most values are biased toward the small side and there are a few outliers. Fortunately, most Neural Network weights follow this distribution in practice, so we can say that quantization like this is appropriate.

2.2. Group Quantization

Pseudo Float Quantization has the advantage of being able to express the distribution of parameters of a neural network well. However, since each parameter has a different coefficient, there is a problem that NPU operations such as Tensor Core, which is operated in group units, cannot be used. NPU has the disadvantage that only specific operations cannot be performed, such as GEMM, but it has the advantage of being able to process the operation very quickly. Therefore, we have improved the existing quantization method to use these calculation devices.

First, since NPU calculation is performed in group units, each calculated group must have the same coefficient value. Therefore, we set the operation unit as a quantization group size and apply SUQ.

Also, for non−linear functions to be processed only with integer arithmetic, all parameters must be expressed only as integers except for common float scale values. For this purpose, we quantized the scale value of each group once more.

$$\mathrm{r} \approx \frac{G}{N} \cdot q_x = \frac{X}{N}\frac{G}{X} \cdot q_x \approx \frac{X}{N} \cdot s_g \cdot q_g \cdot q_x = s \cdot q_g \cdot q_x$$

Above, N means $2^{(bit-1)} - 1$, G and X mean the largest absolute value in the group and total of parameters. We compared the variance of error (α) when quantizing a group's scale value. We confirmed that the error is sufficiently small when the scale values of the group are quantized using the quantization bit applied to each group.

$$Var\left(x - \frac{X}{N} q_x\right) \geq Var\left(x - \frac{X}{N}\left(\frac{G}{X} + \alpha\right) q_g\right) = Var\left(x - \frac{G}{N} q_g - \frac{X}{N} q_g \alpha\right) \frac{1}{3}\left(\frac{X}{N}\right)^2$$
$$\geq \frac{1}{3}\left(\frac{G}{N}\right)^2 + \left(\frac{X}{N}\right)^2 Var(q_g) Var(\alpha)$$
$$\therefore Var(\alpha) \leq \frac{1}{Var(q_g)} \frac{1}{3}\left(1 - \frac{G^2}{X^2}\right) \leq \frac{1}{Var(q_g)} = \frac{1}{Var\left(\frac{x}{G}\right)} \leq \frac{1}{N^2}$$

This method does not accurately express the distribution of parameters, but because quantization is performed for each group, it shows some strength in outliers. In addition, since we quantized the scale value of each group once more, it requires only a bit more memory than the existing scale quantization (about 1/256 of the total when quantized with int8 under the assumption that Tensor Core is used). Since the unit is considered, it is possible to process quickly.

2.3. Error Compensation using Taylor Expansion

The above two methods minimized the error of the weight parameter to improve the accuracy. However, the above methods cannot be applied to non−linear functions without weight parameters. Therefore, we find out how to minimize the error in the non−linear function changed to operate only with Integer arithmetic.

I-BERT([12]) showed that non-linear functions mainly used in Transformer structures could be sufficiently approximated by integer arithmetic. However, since the operation process involves repetitive operations between integers, the risk of overflow may occur. Therefore, there is a limit to how to reduce the error by increasing the quantization bit. In this situation, we tried to reduce the error by using Error Compensation. This has the advantage that it can be easily applied anywhere because there is no need to change the previously presented formula.

Taylor expansion was used to approximate the error caused by quantization. Non-linear functions can be expressed by Taylor expansion as in the equation below, $f'(q_x)$ represents the Jacobian Matrix. Since there is a limit to the values that can be expressed using only integers, all terms corresponding to $O(\epsilon_x)$ are ignored.

$$f(x) \approx f(q_x) + f'(q_x) \cdot \epsilon_x$$

Since derivatives are defined in all non-linear functions, it is possible to calculate the second term using them. However, it does not guarantee that the computation of the second term is simple. In the case of GeLU, Softmax the calculation process was very complicated. On the other hand, in the case of LayerNorm, it was possible to calculate quickly because the Jacobian Matrix could be approximated with a Diagonal Matrix. In addition, since the standard deviation calculation formula in LayerNorm was a section where overflow frequently occurred, it was used to reduce the error.

Chapter 3. Experiments

3.1. Quantization Error

In order to show that the quantization method proposed by us dramatically reduces errors, the errors were measured by quantizing the weights of the GPT−2 model provided by Huggingface. Quantization bit used 4−bit and 8−bit. In the case of PFQ, the exponent is also 4 bits when mantissa is 4−bit, and only a 2−bit exponent term is used to prevent overflow problem when mantissa is 8−bit. It is possible to check the detailed results from the graph in Figure 1.

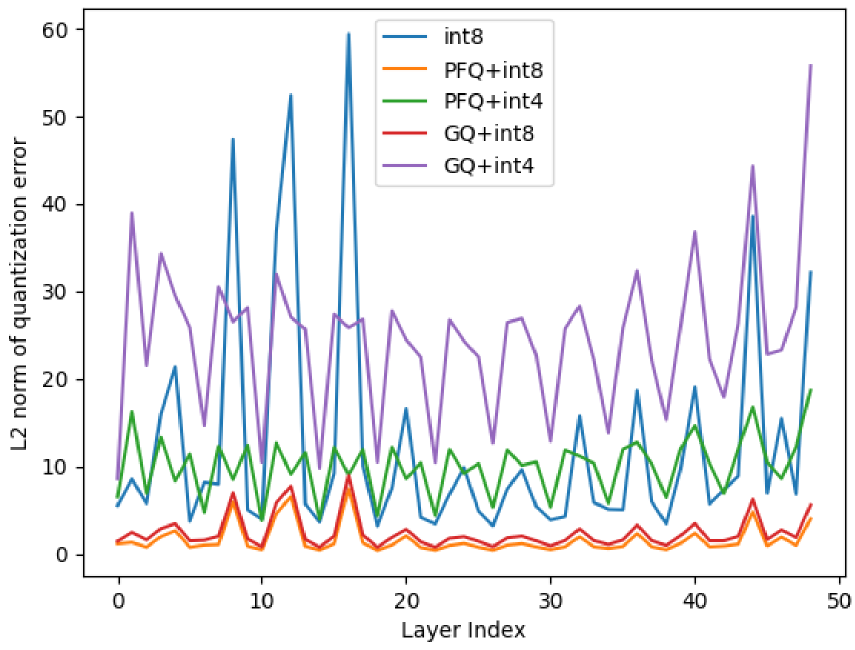

Figure 1. Quantization Error according to quantization method. x−axis means weight of i−th Linear Layer of GPT−2

Looking at the graph, we can see that the proposed method has a lower error than the existing methods. In addition, it can be confirmed that the error is slightly lower when quantized with bits such as PFQ rather than Group Quantization. In particular, when the PFQ method was used with 4−bit, it showed better performance than the 8−bit quantization that occupies the same memory.

Next, we measured the performance of Error Compensation. The IntLayerNorm proposed by I−BERT was used, and the error was measured by changing the quantization bit applied to the input. For the error used for Error Compensation, a 8−bit quantized value was used.

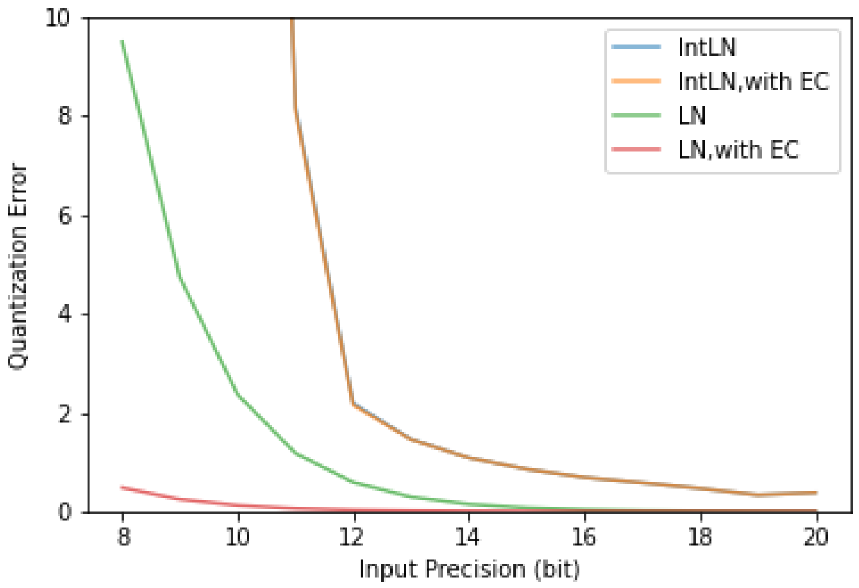

Figure 2. Quantization Error of LayerNorm. There was a good effect was in the original LayerNorm, but no significant effect was shown in the LayerNorm converted to integer arithmetic version.

It was confirmed that the error was significantly reduced when Error Compensation was applied to the original LayerNorm. However, IntLayerNorm, composed of integer arithmetic, had no significant effect. We thought this phenomenon occurred because the Jacobian obtained from IntLayerNorm was inaccurate, and its error was more significant than the error reduction amount by Error Compensation.

Finally, we measured the accuracy of MRPC Task using the Robert-base model. In the case of some methods, it was impossible to implement them optimally, so the accuracy was measured using simulated quantization. Embedding Layer and Activation were implemented using 8-bit Scaling Quantization because it took a long time for quantization, and there were cases where Group Quantization, which is applied due to the size of the group, could not be applied. Therefore, the model size and performance were measured by applying various bits and quantization methods only to all weights of the linear layer. For non-linear functions, the method proposed by

I−BERT was used as it is. In addition, in the case of the experiment below, the performance was measured immediately after proceeding only with quantization without independent learning.

	Precision (bit)	Model Size (MB)	Accuracy w/o Finetuning (%)	Accuracy after Finetuning (%)
Original	32	928.793	87.75	−
PFQ(int8)	10(2+8)	290.522	88.48	87.99
GQ(int8)	8.024	232.814	88.24	87.74
PFQ(int4)	8(4+4)	232.497	87.01	87.99
Int8	8	232.497	87.99	88.23
GQ(int6)	6.018	174.789	87.00	87.99
PFQ(int4)	6(2+4)	174.473	83.82	88.48
Int6	6	174.473	80.64	86.76
GQ(int4)	4.012	116.765	79.41	80.39
Int4	4	116.448	62.01	68.38

Table 1. Comparison of accuracy and quantized model size of roberta−base according to quantization method and precision

From the table above, we can confirm that when the quantization error is small, high accuracy can be preserved without additional training . In particular, it was confirmed that the size of the model is comparable to SUQ, even when Group Quantization was applied. In addition, when less than 8−bit, the difference in accuracy with SUQ began to appear clearly, and when 4−bit was applied, Accuracy of SUQ was reduced to less than 70%, but GQ maintained about 80%.

Chapter 4. Conclusion

In this study, we looked at various ways to improve the performance of Integer−only Quantization. All methods identified were able to reduce errors due to quantization compared to the previous ones. However, some quantization methods were almost impossible to implement in the current software and hardware, and in the case of Error Compensation, there was no significant effect. However, suppose an environment in which this can be implemented is given.

In that case, high−performance improvement can be expected, Error Compensation is also thought to be a good solution to solve the overflow of non−linear operation if additional research is conducted.

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.
[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems,33:1877–1901, 2020.
[3] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Understanding and overcoming the challenges of efficient transformer quantization. arXiv preprint arXiv:2109.12948, 2021.
[4] Insoo Chung, Byeongwook Kim, Yoonjung Choi, Se Jung Kwon, Yongkweon Jeon, Baeseong Park, Sangha Kim, and Dongsoo Lee. Extremely low bit transformer quantization for on-device neural machine translation. arXiv preprint arXiv:2009.07453, 2020.
[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.
[6] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. arXiv preprint arXiv:2208.11580, 2022.
[7] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv:2103.13630, 2021.
[8] Robert M. Gray and David L. Neuhoff. Quantization. IEEE transactions on information theory, 44(6):2325–2383, 1998.
[9] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.
[10] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In 2014 IEEE International Solid-State Circuits

Conference Digest of Technical Papers (ISSCC), pages 10–14. IEEE, 2014.
[11] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2704–2713, 2018.
[12] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In International conference on machine learning, pages 5506–5518. PMLR, 2021.
[13] Zhikai Li and Qingyi Gu. I-vit: Integer-only quantization for efficient vision transformer inference. arXiv preprint arXiv:2207.01405, 2022.
[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.
[15] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks
using logarithmic data representation. arXiv preprint arXiv:1603.01025, 2016.
[16] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5456–5464, 2017.
[17] Gunho Park, Baeseong Park, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. arXiv preprint arXiv:2206.09557, 2022
[18] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision, pages 525–542. Springer, 2016.
[19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei,

Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.
[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
[21] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. arXiv preprint arXiv:2209.13325, 2022.
[22] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks. arXiv preprint arXiv:1802.00150, 2018.

Abstract

Transformer는 다양한 분야에서 최고 성능을 달성한 현재 가장 유명한 인공 신경망 구조이다. 따라서 많은 사람들이 이 모델의 추론이 정수 연산으로만 이루어질 수 있도록 양자화(Integer-only Quantization)를 적용하여 엣지 장치에서도 동작할 수 있도록 하고자 하였다. 하지만, 일반적인 하드웨어의 연산 특징과 비선형 함수의 효율적인 계산을 위해 제약조건이 존재하였고, 때문에 양자화 방법은 크게 달라지지 않았다. 따라서 우리는 Integer-only Quantization에서 사용할 수 있는 새로운 방법들을 연구하여, 기존의 방법들의 한계를 넘을 수 있도록 하였다.