Proceedings of Machine Learning Research vol 245:1–6, 2024 2024 International Conference on Machine Learning and Intelligent Computing

Hierarchical Quantization Algorithm for Deep Learning Network Models

Xiaoqi Mao CHESSICATT1121@163.COM
AnHui Siliepoch Technology Co.,Ltd, HeFei, AnHui, China

Editors: Nianyin Zeng and Ram Bilas Pachori

Abstract

Deep learning network models have achieved inspiring performances in various fields such as computer vision, natural language processing, and biomedicine. However, the high computational and storage costs of the models restrain their application in resource-limited situations. However, due to the increased com-plexity and computation of deep neural networks, there are still some challenges in deploying deep learning models into real-world applications for resource-constrained devices. To address this problem, researchers have proposed various quantization algorithms to decrease the expenditure of calculation and storage in deep learning models. This thesis addresses the problem of hierarchical quantiza-tion of deep learning models and proposes a simple hierarchical quantization al-gorithm that aims to effectively reduce the computation and storage requirements of deep learning network models and maintain the accuracy of the models. To demonstrate the effectiveness of the proposed hierarchical quantization method, we conducted experiments on several classical deep learning models. Our exper-iments prove our approach can better maintain the models' accuracy while reduc-ing the storage and computation requirements compared to the traditional quanti-zation algorithms.

Keywords: deep learning, quantization algorithm, hierarchical quantization, computer vision

1. Introduction

The development of deep learning network models has led to breakthroughs in many applications such as image classification, object detection, and semantic segmenta-tion. However, the operational requirements of deep learning models grow exponen-tially as the size and complexity of the models increase. This makes it difficult to deploy deep learning models in resource-constrained conditions. As a result, light quantization algorithms have become the key to solving this problem.

2. Related works

This section reviews the current mainstream deep learning model quantization techniques, including weight quantization, activation value quantization, and net-work structure quantization, and analyzes their advantages and disadvantages.

According to the stage of applying quantization compression model, the model quantization can be divided into Quantization-Aware Training (QAT), Quantization-Aware Fine-tuning (QAF), Post-Training Quantization (PTQ).

Quantization-Aware Training adds pseudo-quantization operators in the model training process for enhancing the quantized model's accuracy. The statistical input and output data ranges during training, which is suitable for scenarios requiring high model accuracy. The quantization objective is seamlessly embedded into the model's training procedure. This approach adjusted the LLM to representations with low-precision during the training process, enhancing its ability to deal with the loss of ac-curacy caused by quantization processing. This alteration is to keep better perfor-mance after adopting the quantization method. Quantization-Aware Fine-tuning quantizes the LLM during the fine-tuning process for ensuring that the fine-tuned LLM maintains its original performance with a lower bit width. A balance between model compression and performance preservation is achieved by integrating quanti-zation awareness into the fine-tuning. Post-Training Quantization quantizes the LLM's parameters after completing its training. It requires only a small amount of calibration data, and is suitable for scenarios with high ease of use and lack of train-ing resources. The main goal is to cut down the LLM's computational complexity without modifying its architecture. The main advantage of PTO is its simplicity and efficiency. However, PTQ may introduce some degree of accuracy loss in the quanti-zation process.

Post-training quantization can be divided into weight and full quantization.

The former only quantizes the weights of the model to compress the size of the model, and in the inference, the weights are inverted and quantized into the original float32 data, and the subsequent inference process is the same as that of the ordi-nary float32 model. The advantage of weight quantization is that there is no need to calibrate the dataset, there is no need to implement the quantization operator, and the accuracy error of the model is small, because the actual inference is still using the float32 operator, so the inference performance will not be improved.

Full quantization will not only quantize the weights of the model, but also the acti-vation values of the model, and the quantization operator is implemented during model inference to speed up inference where the model requires the user to provide a certain number of calibrated datasets for counting the scores of the activation values at each level. In order to quantize the activation values, cloth and do calibration of the quantized operators. Calibration dataset. Can come from training datasets or input data from real scenarios, the number needed is usually very small.

Park and Lee (2022) et al. improve computational efficiency by quantizing only the weights and optimizing matrix multiplication in LLMs using the BCQ format through to en-hance latency reduction and performance.

Dettmers and Zettlemoyer (2022) et al. use a quantization method with mixed precision decomposition. A matrix decomposition was done first, with 8bit quantization (vector-wise) for the vast majority of weights and activations. A few dimensions of the outlier features are reserved for 16bit, for which high precision matrix multiplication is done.

Lee (2024) et al. introduce a mixed-precision quantization scheme by analyzing how activation anomalies influence weight quantization's error, and give higher precision to weights that are susceptible to activation anomalies.

Dettmers and Alistarh (2023) et al. identify and isolates the anomaly weights, stores them from higher to lower precision, and compresses all other weights into the type of 3-4 bits.

In addition to this, there are full quantization (weight and activation quantization).

Activations in LLM are often complicated by the presence of outliers, and Xiao and Han (2024) et al. address the challenge of quantizing activations. Smooth-Quant observes that different tokens show analogous variations, introducing a scaling transform that effectively smooths the magnitude in a way of channel-by-channel, making the model easier to quantize.

Given the complexity of quantifying activations in LLMs, 6. Yuan and Wu (2023) et al. reveal the challenge of inhomogeneous ranges between different channels, and the problems posed by the presence of outliers. To address this issue, RPTQ quantizes channels by strategically grouping them into clusters, effectively mitigating differ-ences in channel ranges. Moreover, it integrates channel rearrangement into the oper-ations of layer normalization and the weights of linear layer for minimizing the asso-ciated overhead.

While Guo and Zhu (2023) et al. further employ outlier-victim pair (OVP) quantization with low hardware overhead and high performance gain to deal with outliers locally, based on the rule that outliers are more important than the normal value next to a certain one.

Outlier Suppression+ Xiuying Wei and Liu (2023) confirms that the that deleterious anomalies in activation mostly are concentrated in specific channels. Therefore, a new strategy consists of channel-level shifting and scaling operations, which is introduced to correct the anomalies' asymmetric presentation and to reduce the problematic channels' effects. The optimal values of shifting and scaling are quantitatively analyzed, taking into account the asymmetry of the anomalies as well as quantization errors induced by the weights of the next layer.

Wu and He (2023) et al. explore the floating-point (FP) quantization's applicability, with particular attention to the two types of FP8 and FP4. Besides, the work shows that for LLMs, FP8 activations consistently outperform INT8, while for weight quantization, FP4 compares comparably or even superiorly to INT4 in terms of performance. To address the challenges caused by the discrepancy between activations and weights, ZeroQuant-FP requires powering all scaling factors with the number 29 and restricts the factors to a single computational unit. ZeroQuant-FP also embeds a Low Rank Compensation (LoRC) method to further improve its quantization method's validity.

3. Design of hierarchical quantization algorithm

In order to realize hierarchical quantization in deep learning models, this paper proposes a hierarchical quantization algorithm based on gradient information and model structure. The algorithm adjusts the accuracy and distribution of quantization according to the importance of the gradient and the complexity of the model struc-ture by performing hierarchical quantization on different layers of the network.

3.1. Conventional quantization methods

1. int8 (8-bit integer quantization):

Int8 quantization is a widely used algorithm that represents numerical values using 8-bit integers. By reducing the precision from floating-point numbers type of 32-bit to the integers type of 8-bit, int8 quantization significantly reduces memory usage and computational overhead. However, it comes at the cost of reduced accuracy due to the limited range of representable values. Int8 quantization is commonly used in applications where a balance between memory efficiency and accuracy is desired.

2. uint8 (8-bit unsigned integer quantization):

Similar to int8, uint8 quantization represents numerical values using 8-bit integers. The key difference is that uint8 only allows non-negative values, as it utilizes the full range of 8 bits to represent positive values. This quantization algorithm is particularly useful in scenarios where negative values are not expected, such as image processing or signal processing applications.

3. fp16 (16-bit floating-point quantization):

Fp16 quantization is a technique that represents numerical values using 16-bit floating-point numbers. Compared to int8 and uint8 quantization, fp16 provides a higher precision by preserving the decimal part of the number. This makes it suitable for applications that require more accurate calculations, such as scientific simulations or training deep neural networks.

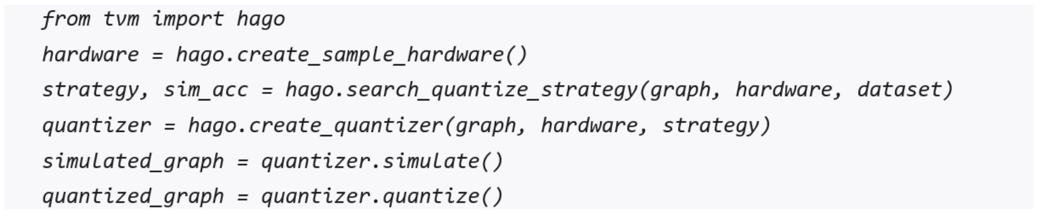
4. bfp16 (bfloat16 quantization):

Bfp16 quantization, also known as bfloat16, is a variation of fp16 that offers a compromise between precision and memory efficiency. The ”b” in bfp16 stands for ”brain,” as this quantization algorithm was initially developed by Google for neural network training. By reducing the precision of the mantissa while preserving the ex-ponent, bfp16 achieves a balance between memory usage and accuracy. It is com-monly used in deep learning frameworks like TensorFlow for improved performance and reduced memory footprint.

3.2. Hierarchical quantization algorithm

1. Calculate the value of loss due to comparative quantization for each layer.

This paper performs the loss calculation after quantization for each layer with the help of TVM.hago API to get the layers which have large loss due to quantization. The procedure is as follows in Figure 1.



```
from tvm import hago
hardware = hago.create_sample_hardware()
strategy, sim_acc = hago.search_quantize_strategy(graph, hardware, dataset)
quantizer = hago.create_quantizer(graph, hardware, strategy)
simulated_graph = quantizer.simulate()
quantized_graph = quantizer.quantize()
```

Figure 1: Calculation of loss using API.

2. Individually quantize the layers with larger losses.

According to the hardware operation requirements, the quantization type is select-ed to quantize the model, and after obtaining the layers with large quantization loss, ”sub-quantization”, i.e., high-precision quantization, is performed on these layers. In this paper, we take the int8 quantization type as an example, and in the layer where the error value is larger, we quantize it with 16-bit bits in order to retain a higher preci-sion.

3. Connect each network layer after quantization.

After iterative attempts, we determine the optimal choice of quantization for each layer and output this hierarchical quantization model.

4. Experiments and results analysis

This section verifies the effectiveness of the hierarchical quantization method by conducting experiments on common image classification datasets. The experiment results show that the proposed algorithm can maintain classification accuracy while reducing computational and storage requirements.

This experiment uses shufflenetv1, three of the more classical models, in the five quantization cases of uint8, int8, fp16, bfp16 and our method. The coco dataset is used for accuracy testing and performance testing.

Table 1: Table of experimental accuracy results.

Model	Uint8	Int8	Fp16	Bfp16	cOur method
Shufflenetv1	0.5818	0.6024	0.6316	0.6324	0.6569
Resnet50	0.7544	0.7516	0.7584	0.7582	0.7683
googlenet	0.6646	0.6736	0.6814	0.6802	0.6972

From the results in Table 1, we can see that by utilizing the hierarchical quantization algorithm, the accuracy loss of the model is also reduced while the computational cost of the model is reduced. The layered quantization algorithm in this paper has high practical application value and good optimization effect.

5. Conclusion

This paper discussed the advantages and limitations of quantization algorithms and explored some challenges and solutions in quantization of deep learning models to further improve the application of deep learning models in resource-constrained environments.

The proposed hierarchical quantization algorithm provides an effective solution for the application of deep learning models in resource-constrained environments. The algorithm achieves the goal of maintaining model accuracy while reducing com-putation and storage requirements by intelligently adjusting the quantization accura-cy and distribution. However, there are still some challenges that need to be further investigated and addressed.

The future research direction can be carried out in the following aspects. First, we can explore adaptive quantization algorithms, i.e., dynamically adjusting the quanti-zation accuracy during the network training process according to the characteristics of the network and the task requirements, in order to achieve more precise model representation and higher task accuracy. Second, researchers can focus on the opti-mization of quantization algorithms on different hardware platforms to achieve more efficient model inference and deployment. In addition, combining deep learn-ing hierarchical quantization algorithms with other optimization techniques, such as joint optimization of pruning and quantization, can further improve model perfor-mance and efficiency. In conclusion, future research directions in deep learning hier-archical quantization algorithms will revolve around adaptive quantization, cross-platform optimization and combination with other optimization techniques to achieve more efficient, flexible and accurate deep learning model applications. Re-search in this field will provide important support for the development and applica-tion of artificial intelligence technology.

References

Lewis M. Belkada Y. Dettmers, T. and L Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022.

Svirschevski R. A. Egiazarian V. Kuznedelev D. Frantar E. Ashkboos S. ... Dettmers, T. and D Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *ArXiv*, abs/2306.03078, 2023.

Tang J. Hu W. Leng J. Zhang C. Yang F. ... Guo, C. and Y Zhu. Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23. ACM, June 2023. doi: 10.1145/3579371.3589038.

Jin Jungyu Kim Taesu Kim Hyungjun Park Eunhyeok Lee, Changhun. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models, 2024.

Park B. Kwon S.J. Kim B. Lee Y. Park, G. and D Lee. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *ArXiv*, abs/2206.09557, 2022.

Yao Z. Wu, X. and Y He. Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats, 2023.

Lin J. Seznec M. Wu H. Demouth J. Xiao, G. and S Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024.

Yuhang Li Xiangguo Zhang Ruihao Gong Jinyang Guo Xiuying Wei, Yunchen Zhang and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling, 2023.

Niu L. Liu J. Liu W. Wang X. Shang Y. Sun G. Wu Q. Wu J. Yuan, Z. and B Wu. Rptq: Reorder-based post-training quantization for large language models, 2023.