DQ-CLIP: Post-Training Quantization for Dynamic Quantized CLIP

Feiyang Chen[*] `fychen@ucla.edu` **Yadi Cao**[*] `yadicao95@ucla.edu` **Zhaoqian Wang**[*] `zhw173@ucla.edu`

Department of Computer Science
University of California, Los Angeles

Dec 9, 2022

Abstract

Our experience of the world is multimodal (vision, language, audio, etc). Although researchers have spent much effort on model compression to reduce the huge memory consumption and computational power of increasingly large models, how to compress multimodal models, especially Vision-Language Pretraining (VLP) models, is still under-explored. We know pruning and distillation can reduce the size of machine learning models, but typically require retraining with the same or even more computational resources. In contrast, post-training quantization incurs negligible costs. In this project, we aim to study the potential for reducing the size and computational complexity of a multimodel model (eg. CLIP) using post-training quantization without incurring significant additional computation. Our experiments showed that quantization has several advantages, including improved performance for devices with limited computational power. One of our key findings is that combined quantization of both the image and text modules leads to the best model size reduction, resulting in a 41% reduction in model size with only a moderate drop in accuracy. Our code is available at here.

1 Introduction

In recent years, transformer-based architectures have achieved competitive performance in a variety of machine learning applications, including natural language processing (NLP) and computer vision (CV). These models are often pre-trained on large datasets and fine-tuned for downstream tasks. However, despite their excellent performance, pre-trained transformers are very large, with some having billions of parameters and more floating-point operations (FLOPs) than traditional models such as convolutional neural networks (CNNs) [1]. As a result, it can be difficult to deploy pre-trained transformer models in latency-sensitive applications or in resource-limited environments such as edge devices with limited computational power and memory.

With the success of deep learning models in NLP and CV, researchers have become increasingly interested in solving more complex problems involving multiple data modalities [1]. The real world contains a vast amount of multimodal information, with vision and language being the two

[*]Equal contribution.

most representative modalities. Researchers have focused on model compression to reduce the significant memory consumption and computational power of increasingly large models. However, the compression of multimodal models, particularly image-text models, remains an under-explored area. In this project, we aim to investigate the use of network quantization for compressing a representative multimodal model: Contrastive Language-Image Pre-training (CLIP)[2].

2 Related Works

2.1 Multi-modal Model

With the undeniable success of transformer-based models across tasks in different domains, researchers have begun to turn their attention to solving more complex machine learning problems that involve multiple data modalities such as image retrieval[2], image captioning[3], and image synthesis[4]. One notable example is the CLIP network[2] that takes advantage of natural language supervision to learn a more efficient image representation. As shown in Figure **??**, CLIP trains an image encoder and a text encoder simultaneously to learn a contrastive representation. More concretely, given N image-text pairs, CLIP jointly learns the image and text embeddings that maximize the cosine similarity of the N correct image-text embedding pairs while minimizing N^2-N incorrect pairs. At inference, the text encoder of CLIP converts labels into prompts and computes the encoding of each prompt. The image encoder computes the encoding of the image and the image-text embedding pair with the highest cosine similarity is chosen as the prediction of the model. CLIP demonstrated excellent zero-shot transferability and achieved 76.2% top-1 accuracy on the ImageNet dataset without seeing any of its labels at training.

2.2 Model Compression

2.2.1 Pruning

Pruning is a popular way to reduce the dimensions of vision transformers [5]. The key idea behind pruning is to allocate an importance score for each dimension, and discard a large number of dimensions with small importance scores in order to achieve a high pruning ratio without significant loss in accuracy. Meanwhile, dimensional redistribution can be applied together with the pruning process to achieve better performance [6]. It is also notable that the model after effective pruning may achieve better performance than the original model [7].

2.2.2 Knowledge Distillation

Knowledge distillation is a model compression method in which a small model (student model) leverages soft labels from a large model (teacher model) for transferring knowledge [8]. The soft labels from the teacher are well known to be more informative than hard labels and lead to better student training ([9, 10]). [11] introduced a new distillation procedure based on a distillation token for vision transformer [12], which plays the same role as the class token, except that it aims at reproducing the label estimated by the teacher. Both tokens interact in the transformer through attention. This transformer-specific strategy outperforms vanilla distillation by a significant margin.

2.2.3 Quantization

Quantization is a widely adopted technique to enable efficient inference. By quantizing the network into low-bit representation, the inference of the network requires less computation and less memory, while still achieving little performance degradation. A crucial point in this line of work is to determine the clipping range for the weights. [13] determines the clipping range by considering all of the weights in convolutional filters of a layer, while [14] quantizes the Transformer using groupwise quantization. To remedy the accuracy loss induced by quantization, researchers also propose Quantization-Aware Training, in which the usual forward and forward pass are performed on the quantized model in floating point, but the model parameters are quantized after each gradient update.

3 Methodology

Generally, the goal of quantization is to reduce the precision of both the parameters (θ) and intermediate activation maps to low-precision (e.g., 8-bit integer), with minimal impact on the generalization performance of the model. To do this, we need to define a function that can quantize weights and activations to a finite set of values. A popular choice for a quantization function is as follows:

$$Q(r) = \text{Int}(r/S) - Z, \tag{1}$$

where Q is the quantization mapping function, r is a real-valued input (i.e., weights, activations), S is a real-valued scaling factor, and Z is an integer zero-point. This function maps a real value r to some integer value. This method is known as uniform quantization, since the resulting values are uniformly spaced. There are also non-uniform quantization methods. It is also possible to recover the real value r from its quantized value $Q(r)$ through dequantization:

$$\tilde{r} = S(Q(r) + Z) \tag{2}$$

The value of the recovered $\tilde{r}$ may not be exactly the same as r due to the rounding error in quantization. The most important aspect of quantization is determining the value of the scaling factor S. This scaling factor essentially divides real values r into a number of partitions:

$$S = \frac{\beta - \alpha}{2^b - 1} \tag{3}$$

where $[\alpha, \beta]$ denotes the clipping range and b is the quantization bit width. To determine the value of S, the clipping range $[\alpha, \beta]$ must be defined first. The process of choosing the clipping range is often referred to as calibration. One straightforward approach is to use the minimum and maximum of the input for this clipping range, i.e., $\alpha = r_{\min}$ and $\beta = r_{\max}$. This is an asymmetric quantization scheme, since $-\alpha \neq \beta$. It is also possible to use a symmetric quantization scheme, e.g., $-\alpha = \beta = \max(|r_{\max}|, |r_{\min}|)$. In this case, the quantization function in Eq. 1 can be simplified by setting $Z = 0$.

3.1 DQ-CLIP

Quantization methods can be divided into two categories: Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ). QAT depends on training to achieve aggressively low-bit (e.g.

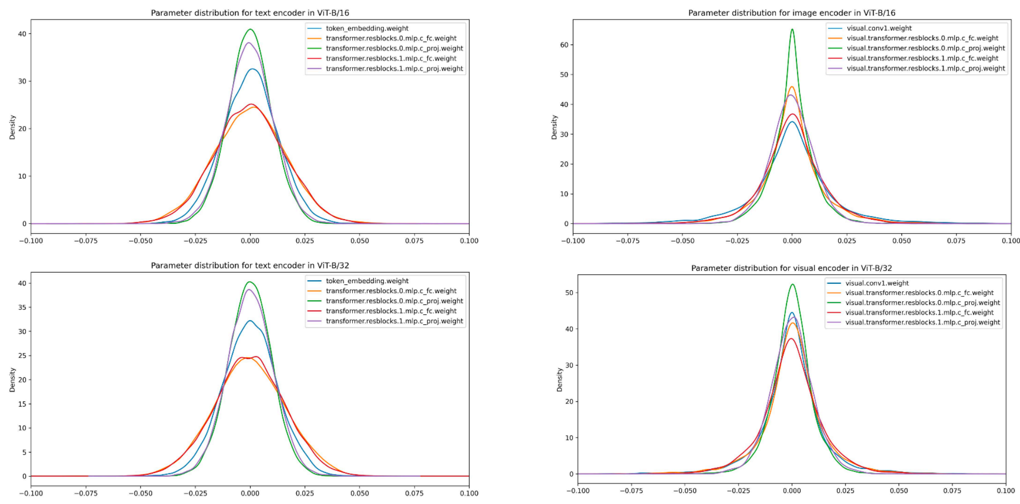

Figure 1: Parameter Distribution in CLIP

2-bit) quantization and promising performance, while it often requires high-level expert knowledge and huge GPU resources for training or fine-tuning. To reduce the above costs of quantization, PTQ, which is training-free, has received more widespread attention and lots of excellent work arise. So in this project, we focus on the PTQ method.

The parameter distributions of some important modules in CLIP are shown in Figure 1. The densely distributed parameters make quantization a suitable approach. The key idea behind dynamic quantization is that the scale factor for activations is determined dynamically based on the observed data range at runtime. This ensures that the scale factor is optimized to preserve as much of the signal in the data as possible. The model parameters, on the other hand, are known during model conversion and are converted ahead of time and stored in INT8 form. Arithmetic in the quantized model is performed using vectorized INT8 instructions. The below equations show the details of quantized multiplication arithmetic. Since the scale factors of input x, weight y, and output z are all known for a given neural network, they can be pre-computed before network forwarding. Accumulation is typically performed using INT16 or INT32 to avoid overflow. This higher precision value is scaled back to INT8 if the next layer is quantized, or converted to FP32 for output. Dynamic quantization has relatively few tuning parameters, making it well suited for use in production pipelines as a standard part of converting CLIP models for deployment.

4 Experiments and results

4.1 Linear probe evaluation.

We applied dynamic quantization to two different ViT models: ViT-B16 and ViT-B32 (Table. 1), which are the image models of CLIP. The purpose of this experiment was to test the basic specifications, such as the compression ratio and the reduction in inference time, of quantization. We further used the quantized models in a linear regression experiment to assess the accuracy drop

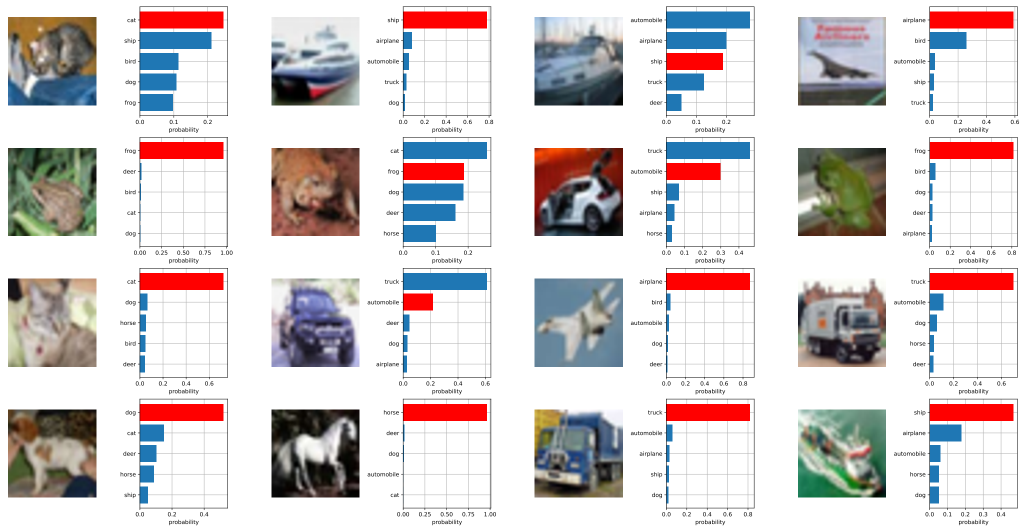

Figure 2: Example of the image-text retrieve test

CLIP Model	Component	Data Type	Model Size [MBs]	Classification Accuracy [%]	Inference time [ms]
ViT-B/32	-	-	577.18	79.69	0.2061
ViT-B/32	Image	int8	415.20	77.21	0.2114
ViT-B/16	-	-	570.86	82.31	0.2073
ViT-B/16	Image	int8	408.88	67.76	0.2083

Table 1: Detailed results of the linear probe test.

in the image models. The results showed that ViT-B32 is a significantly better solution than ViT-B16. We also found that quantizing image modules contributed more to both size reduction and accuracy drop. The inference time was not affected by quantization in PyTorch, but it is not clear whether this would be the case when deployed on resource-limited devices.

CLIP Model	Component	Data Type	Model Size [MBs]	Top-1 Accuracy [%]	Top-5 Accuracy [%]
ViT-B/32	-	-	577.18	89.74	99.55
ViT-B/32	Combined	int8	343.22	84.98	99.08
ViT-B/32	Image	int8	415.20	86.88	99.06
ViT-B/32	Text	int8	505.20	88.51	99.39
ViT-B/16	-	-	570.86	90.83	99.44
ViT-B/16	Combined	int8	336.90	56.13	91.61
ViT-B/16	Image	int8	408.88	61.60	94.64
ViT-B/16	Text	int8	498.88	89.75	99.23

Table 2: Detailed results of the zero-shoot image-text retrieve test.

4.2 Zero-shoot image-text retrieve test

We conducted a zero-shot image-text retrieval test, in which the images were from the test dataset and had not been seen by the models (Fig. 2). The text was generated by fitting the 100 possible

a person looking at a camera on a tripod	0.30	0.19	0.21	0.20	0.21	0.16	0.14	0.19
a portrait of an astronaut with the American flag	0.17	0.28	0.22	0.16	0.17	0.15	0.15	0.13
a rocket standing on a launchpad	0.20	0.19	0.30	0.17	0.18	0.16	0.14	0.17
a black-and-white silhouette of a horse	0.21	0.11	0.15	0.35	0.15	0.17	0.15	0.17
a facial photo of a tabby cat	0.16	0.12	0.12	0.15	0.31	0.12	0.17	0.12
a red motorcycle standing in a garage	0.12	0.15	0.16	0.16	0.15	0.32	0.13	0.14
a cup of coffee on a saucer	0.17	0.15	0.12	0.15	0.18	0.12	0.29	0.14
a page of text about segmentation	0.20	0.15	0.16	0.20	0.20	0.16	0.20	0.35

(a) original model

0.26	0.19	0.20	0.21	0.22	0.19	0.19	0.22
0.20	0.28	0.23	0.22	0.20	0.21	0.20	0.19
0.22	0.23	0.28	0.19	0.20	0.21	0.20	0.22
0.21	0.20	0.18	0.32	0.20	0.22	0.21	0.21
0.21	0.19	0.17	0.20	0.28	0.20	0.20	0.18
0.17	0.20	0.20	0.20	0.18	0.30	0.19	0.17
0.19	0.20	0.17	0.19	0.21	0.18	0.28	0.17
0.21	0.20	0.20	0.22	0.21	0.21	0.21	0.26

(b) ViT-B32, quantized on both image and text

Figure 3: Similarity between the image and text embedding; compared between the full model (left), and our best-quantized candidate (right).

class labels into two different templates (one with seven prompt templates and the other with 80). In this experiment, ViT-B16 with text-only quantization preserved the most prediction accuracy compared to the base model. However, it only reduced the model size by 13%. ViT-B32 had the best trade-off, reducing the size by 44% while only reducing the accuracy by 5% (Table. 4.1).

4.3 Image-test encoding similarity

We picked the best model (ViT-B32 with combined quantization) and calculated the cosine similarity between the embedding of unseen images and text (Fig. 3). The similarity reflects how well the two models contrast the correct text-image pairs. Our results showed that the similarity between the original full model and our best candidate was very similar, indicating that the quantization process successfully preserved the key features of the contrastive model while removing redundant information from the model.

5 Conclusion and Future Works

In this project, we have investigated the potential for reducing the size and computational complexity of a multi-model (CLIP) without incurring significant additional computation. Our experiments with post-training quantization have shown several advantages. For instance, we have found that quantizing ViT-B16 typically leads to poorer performance compared to ViT-B32. In terms of inference time, there is little difference between the full model and the quantized model when using PyTorch. However, the difference becomes more significant when storing the model as a PyTorch Script and loading it as a JIT model, indicating that quantization is particularly beneficial for devices with limited computational power. One of our key findings is that combined quantization of both the image and text modules leads to the best model size reduction. In our experiments, quantizing the ViT-B32 model with both models resulted in a 41% reduction in model size, albeit with a moderate drop in accuracy (2 to 4%). Further research into finding the optimal quantized MHA models could also be worthwhile.

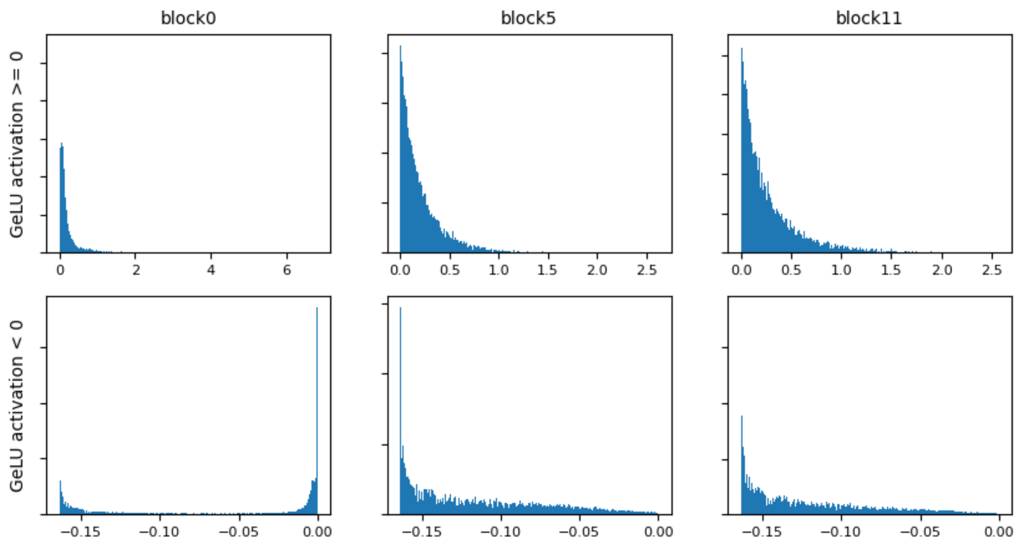

Figure 4: The distribution of GeLU activations. The asymmetrical distribution indicates that it is not suitable for quantization.

References

[1] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on vision transformer. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1):87–110, jan 2023. 1

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1, 2.1

[3] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning, 2019. 2.1

[4] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. 2.1

[5] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. arXiv preprint arXiv:2104.08500, 2021. 2.2.1

[6] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. arXiv preprint arXiv:2110.04869, 2021. 2.2.1

[7] Hao Yu and Jianxin Wu. A unified pruning framework for vision transformers. arXiv preprint arXiv:2111.15127, 2021. 2.2.1

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015. 2.2.2

[9] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986, 2021. 2.2.2

[10] Longhui Wei, An Xiao, Lingxi Xie, Xiaopeng Zhang, Xin Chen, and Qi Tian. Circumventing outliers of autoaugment with knowledge distillation. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, pages 608–625. Springer, 2020. 2.2.2

[11] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In International Conference on Machine Learning, pages 10347–10357. PMLR, 2021. 2.2.2

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 2.2.2

[13] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342, 2018. 2.2.3

[14] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821, 2020. 2.2.3

A Appendix

A.1 Failed experiments

In addition to the above experiments, we also attempted several experiments that ultimately failed.

Fully-Distillation CLIP, Distillation-assisted-Pruning combined CLIP. We implemented pure distillation and distillation-assisted pruning for the CLIP model, but had to conduct retraining on a much smaller dataset due to hardware limitations. This resulted in a near-random-selection zero-shot top-1 accuracy and a drop of approximately 15% in top-5 accuracy, indicating that distillation of the CLIP model must be performed on the original dataset, making it infeasible in our case.

Static Quantization on CLIP:ViT models. we also attempted to include additional ablations for the quantization experiments, but these were unsuccessful. We attempted to implement static quantization of the CLIP:ViT model, but were unable to successfully do so due to the multi-head-attention(MHA) not supporting static quantization. We attempted to solve this problem by creating a custom MHA that is quantizable and copying all the weights from all the MHAs in the pre-trained CLIP:ViT model to the custom MHA using corresponding (but not necessarily the same) keys. However, this approach was ultimately unsuccessful because the following GeLU function is neither supported nor suitable for quantization (Fig. 4). If we convert the data before or after gelu, this hurts computational efficiency and needs extra effort to tweak.

Static Quantization on CLIP:RESNET models We also attempted static quantization of the CLIP:RESNET model, fusing the batch normalization layer into the convolution layer since batch normalization is not supported by static quantization. However, we encountered a problem where the reduced model based on the original implementation hindered downstream tasks due to a data type issue, and ultimately stopped this trial due to the complex implementation and the fact that CLIP:RESNET is no longer state-of-the-art.