Report on Model Compression

He Zhuwei 3036373348 Jiang Shuhao 3036374213
Zhao haotian 3036436932 Lu Tianjian 3036437326

1 Introduction

While deep learning has achieved tremendous success in various fields, it faces increasingly severe challenges in terms of model size and computational requirements. Taking ChatGPT as an example, GPT-2 has storage requirements of several GB, followed by GPT-3 requiring 800GB of storage; for GPT-4, demanding several TB of storage. This is unacceptable, which has led to the emergence of model compression techniques. Through model compression, we can reduce storage and computational requirements while maintaining model performance as much as possible, making it more feasible to deploy models on resource-constrained devices.

Currently, common compression methods include pruning, quantization and knowledge distillation. In our project, we will train an image recognition model on the CIFAR-10 dataset, then compress the model using pruning, quantization, and knowledge distillation methods, and compare the model performance and size before and after compression.

2 Methods

2.1 Pruning

2.1.1 Sparsification

Sparse models are more conducive to pruning, thus sparsification before pruning can effectively improve pruning efficiency. Sparsification works by imposing constraints on weights, making most weights zero or approaching zero, thereby effectively reducing redundant parameters in the model. Common sparsification techniques include L1 regularization, L2 regularization, Dropout, etc., and we adopt L1 regularization in our approach. L1 regularization encourages the model to learn smaller parameters by adding the absolute values of weights to the loss function.

$$L_{\text{total}} = L_{\text{original}} + \lambda \sum_i |w_i|$$

Sparsification prepares smaller weights for pruning, providing convenience for pruning, and making pruning easier to identify redundant structures or neurons. Meanwhile, storing sparsified matrices can save significant storage space, as shown in Figure 1, since sparse matrices only require non-zero data and their positions. However, the storage of sparsified matrices is not our focus; we only use sparsification to generate weights.

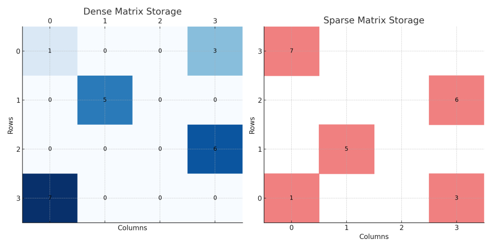

Figure 1: Sparsification

2.1.2 Magnitude-based pruning and fine-tuning

Pruning is based on the principle of removing unimportant weights, neurons, or even layers to reduce the complexity of the model and thus reduce storage requirements. According to whether it significantly affects the model structure, pruning is divided into unstructured pruning and structured pruning. In our project, we adopt the magnitude-based pruning method which belongs to unstructured pruning.

$$\text{Pruning criterion: } |w_i| < \theta, \text{ where } \theta \text{ is the selected threshold value}$$

However, the selection of the threshold value θ is crucial. If θ is too large, important weights may be pruned, resulting in model performance degradation. Conversely, if θ is too small, the pruning effect is not significant. This is why we need to perform fine-tuning - after pruning, fine-tuning can help recover some of the lost weights or further discard unimportant weights.

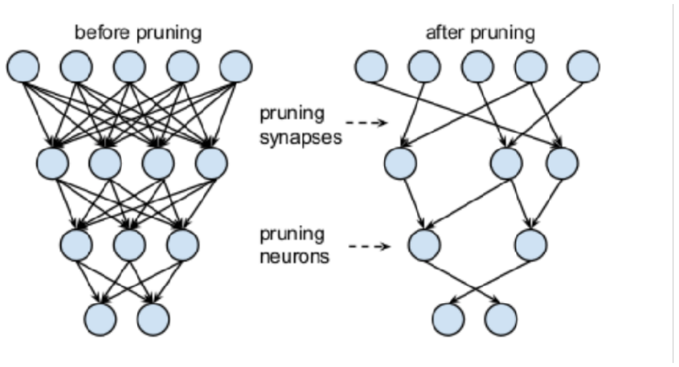

Figure 2: Effect of Pruning

2.2 Quantization

Quantization is a technique that stores parameters using lower precision data types. Typically, models use float32 variables by default to store parameters, while lower precision uses float16 or int8, and in extreme cases, even 1-bit storage can be used. Here, we take the compression from float32 to int8 as an example, which is also a commonly used quantization method. The value range of float32 is $[-2^{31}, 2^{31}-1]$, occupying 4 bytes, while int8 has a range of $[-2^7, 2^7-1]$, occupying 1 byte. The essence of quantization is mapping from a larger interval to a smaller interval. Quantization can be applied to both weights and activation values, and the process generally involves scaling followed by rounding. First, we determine the scaling factor, defined as:

$$\Delta = \frac{\max(a) - \min(a)}{2^b - 1}$$

If it is linear quantization, the quantization compression process can be represented as:

$$\hat{a} = \text{round}\left(\frac{a - Z}{\Delta}\right) \tag{1}$$

Z is the zero point of the original data. If Z is 0, it is symmetric quantization.

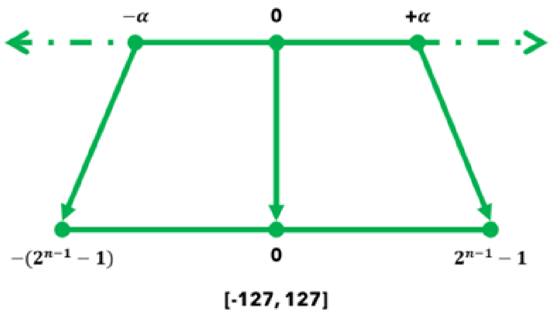

Figure 3: Effect of quantization

If Z is not 0, it is asymmetric quantization, and Z's meaning is to maintain the symmetry of the compressed model. In the inference stage, the quantized data is restored to floating-point calculation, which is called dequantization, but due to the rounding operation, there is inevitably a loss of precision.

$$a_{\text{rec}} = \Delta \times \hat{a} + Z \tag{2}$$

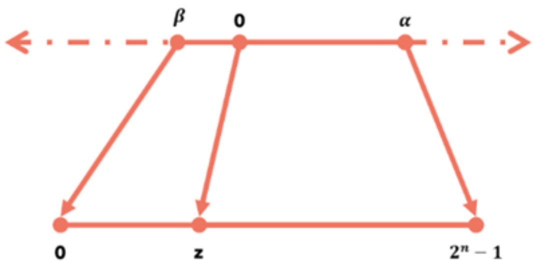

Figure 4: Asymmetric quantization

2.3 Knowledge Distillation

The essence of knowledge distillation is to transfer knowledge from a teacher model to a student model, maintaining the teacher model's high performance while keeping the student model's storage requirements and computational efficiency low. Traditional deep learning optimizes parameters by minimizing the error between model output and ground truth using hard labels, whereas knowledge distillation uses soft labels.

The distillation process can be divided into two stages: training the teacher model and using it to generate soft labels for input data; training the student model by minimizing the difference between its predictions and the teacher model's soft labels, aiming to approximate the teacher model's distribution as closely as possible. The difference between the student model's predictions and the teacher model's soft labels is called the distillation loss:

$$L_{\text{distill}} = -\sum_i q_t(i) \cdot \log\left(\frac{q_s(i)}{T}\right) \tag{3}$$

The difference between the student model's predictions and hard labels is the traditional cross-entropy loss:

$$L_{\text{hard}} = -\sum_i y(i) \cdot \log(q_s(i)) \tag{4}$$

The final loss function is:

$$L_{\text{total}} = \lambda L_{\text{distill}} + (1 - \lambda) L_{\text{hard}} \tag{5}$$

λ is the weight of the distillation loss, which determines the trade-off between model performance and efficiency.

3 Experiment and Analysis

3.1 Target Model for Compression

CIFAR-10 is a dataset of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We first train VGG16 model based on CIFAR-10, and finally achieve an accuracy of 89.9%,and the model size is 152.94MB. This model predicts each image in 5.8ms. All model compression methods in our experiment are based on this initial model.

3.2 Analysis of Pruning

First, we perform sparsification on the model. And after sparsification, the model test accuracy reaches 90.2%, and the model size is 152.94MB. This is because sparsification encourages the model to generate smaller weights, possibly avoiding overfitting. Need to emphasize that we only use sparsification to generate low-weight models, not storing them in sparse matrix form, so the storage size remains unchanged.

Then, we perform pruning with pruning ratio 0.7 on the model. Before fine-tuning, the model accuracy drops to 66.9% and the model size is reduced to 8.67MB. Although there is a significant reduction in storage size, this level of performance degradation is clearly unacceptable. Therefore, we conduct multiple rounds of fine-tuning to attempt to recover some of the lost weights. Finally, our model achieves an accuracy of 87.4% with a storage size of 17.31MB.

Although the storage size increased by nearly 9MB, the accuracy improved from 66.9% to 87.4%, very close to the original model, indicating that we have successfully recovered some of the unimportant weights. We significantly improved the model's prediction accuracy at the cost of a relatively small increase in storage size, demonstrating that fine-tuning is a necessary process for pruning. The prediction time for each image is 5.7ms, which is almost identical to the original model. This is because we used unstructured pruning, which does not change the depth of the model.

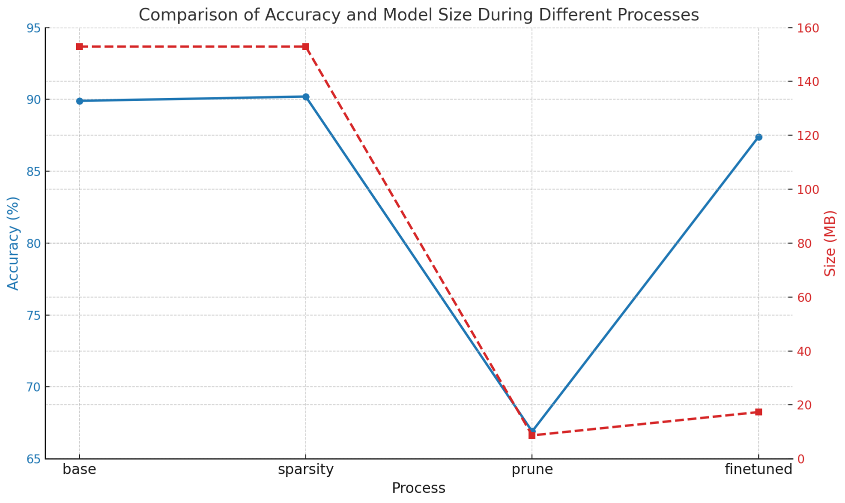

Figure 5: Model accuracy and size during pruning

Since the fine-tuned model reconstructs the weights that disappeared during the pruning process, the model performance after fine-tuning is similar across different pruning ratios. Therefore, we investigate the changes in model accuracy and size before fine-tuning under different pruning ratios. As shown in Figure 6, before fine-tuning, the storage size of the model decreases with the increase of the pruning ratio, which is consistent with the characteristics of model compression. Before the pruning ratio is 0.65, the accuracy remains unchanged, because the weights removed in this range are not enough to affect the prediction performance. When the pruning ratio exceeds 0.65, the prediction accuracy of the model begins to drop significantly, from 72.63% to 49.8%. Although fine-tuning can recover some accuracy, the training time for fine-tuning itself will also increase. Therefore, in the process of selecting the pruning ratio, it is necessary to balance the storage size and model performance.

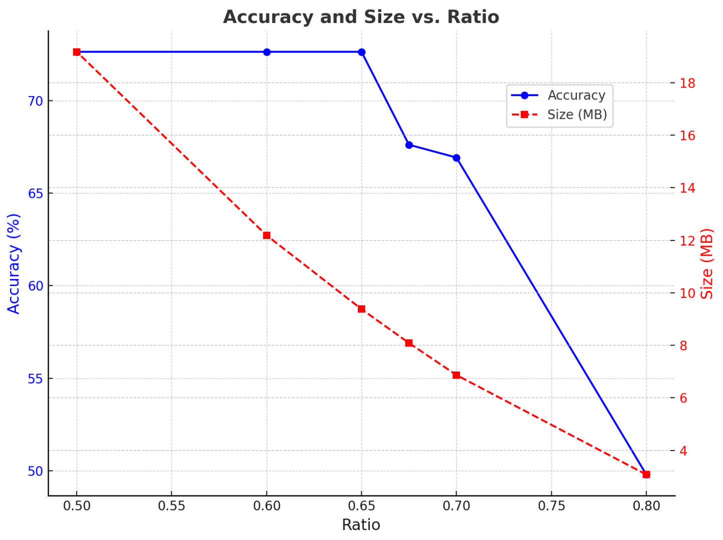

Figure 6: Model accuracy and size before fine-tuning

3.3 Analysis of Quantization

In practical applications, quantization is usually performed to 8-bit and the performance of 4bit or2-bit quantization will drop significantly. Therefore, we mainly perform experiments on the quantization of 16-bit and 8-bit. We first implement 16-bit quantization. After quantization, the model accuracy drops to 86.6%, and the model size is reduced to 52.23MB. This storage size is significantly larger than that of pruning, because quantization retains the weights of all layers in the model, while pruning only retains the weights of the important layers. The model's accuracy only decreased by 3.32%, which is similar to the results after pruning and fine-tuning, but the storage space is twice that of pruning. This indicates that model quantization performs worse than pruning in compressing VGG16 on the CIFAR-10 dataset. The prediction time for a single image is 3.09 seconds, which is about 2ms less than the original model. This is because the reduced precision leads to lower computational power requirements.

Model	Accuracy (%)	Size (MB)	Time (ms)
Base	89.9	152.94	5.8
Quantization (16-bit)	86.6	52.23	3.09
Quantization (8-bit)	81.4	26.12	1.57

Table 1: Comparison of Different Quantization Methods

We further consider the quantization of 8-bit. After quantization, the model accuracy drops to 81.4%, and the model size is reduced to 26.12MB. The model size is almost half of the 16-bit version, which aligns well with the nature of quantization - as each data point is quantized from 16-bit to 8-bit storage, the storage space correspondingly reduces by half. After 8-bit quantization, the prediction time for each image is 1.57 seconds, which is 1.52 seconds faster than the 16-bit quantization.

3.4 Comparison with Knowledge Distillation

Knowledge distillation does not perform exceptionally well in this case, with a prediction accuracy of only 84.8% and a student model size of 18.1MB. The possible reason is: due to the overly complex network structure of the original model, it becomes difficult for the student model to learn the knowledge from the teacher model at a lower cost during the training process.

Methods	Base	Pruning	Fine-tuned	Quantization(16-bit)	Distillation
Accuracy	89.9%	66.9%	87.4%	86.58%	84.8%
Size (MB)	152.94	8.67	17.31	52.23	18.1

Table 2: Comparison of Different Model Compression Methods

4 Conclusion

Pruning, quantization, and knowledge distillation all show good effects on the CIFAR-10 image recognition model. Among them, pruning demonstrates the most outstanding performance, maintaining the highest prediction accuracy while achieving the smallest storage size. This compression effect is very useful in practical applications, especially in embedded devices and mobile devices. We mainly summarize the application directions of pruning and quantization.

Although we chose unstructured pruning in our project, it should be noted that pruning mainly changes the network structure, thereby achieving hardware acceleration effects, and is more important in fields that require real-time feedback, such as navigation, simultaneous translation, and autonomous driving. Quantization mainly reduces the storage size of the model, which is more important in the field of cloud computing.

The main characteristic of quantization is the change in numerical precision, which reduces computational complexity and processing time, though some loss in accuracy is inevitable. This compression principle is particularly suitable for hardware with lower numerical precision capabilities, such as TPUs and FPGAs. It is mainly applied in fields where precision loss is less critical, such as IoT (Internet of Things) and embedded systems. For these application areas, the simplicity of model operations takes priority over precision guarantees.