CINNAMON WEEK 5 HOMEWORK REPORT – ĐINH NGỌC ÂN

Introduction

In this report, I would like to present my report on my findings on compressing the model trained on CIFAR100 Dataset. Compressing the model is a necessary step if you want to reduce the number of computations in a model while maintaining the accuracy on the validation data.

A compressed model can help run on weaker hardware and perhaps CPU, as well as Raspberry Pi devices to allow for breakthrough on robots where their small memory can store a really powerful model like ResNet50, etc.

There are 4 main types of model compression:

- Model distillation (using teacher-student training)
- Quantization
- Pruning
- Low-Rank

I will be presenting my report on **Quantization** and **Pruning** as I have conducted research into the two methods in the short time doing the homework.

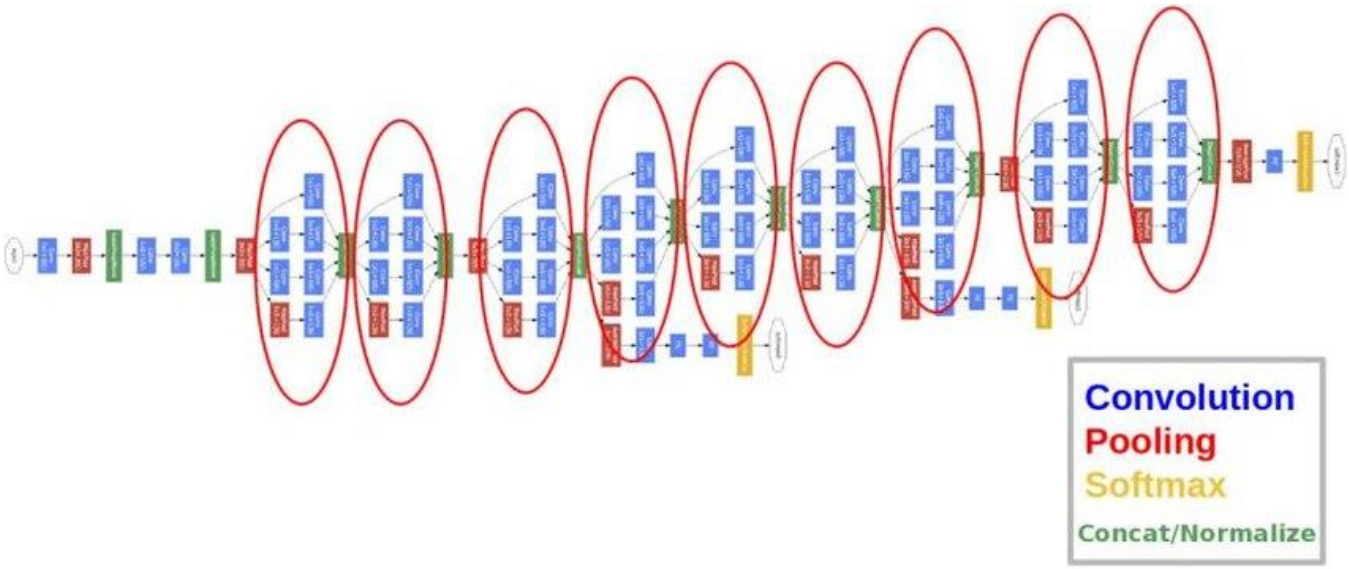
Information

Hardware used:

- Platform: Kaggle and Google Colab.
- Card: Nvidia Tesla T4. (16GB)
- Language: Python with PyTorch 2.0.0
- CUDA: 11.4

Models used:

- GoogleNet. (Because I like that it has “Google” in the name)



- Number of parameters: 6.4M.

Compression Techniques used:

- **Main**: Structured Pruning with 20% sparsity rate.
- **Others**: Post Training Static Quantization (for CNNs).

Training Details – Base Model

- The model is trained on the CIFAR100 Dataset.
- 10000 images are used for validation.
- Number of epochs: 200.
- Initial Learning rate: 0.1.
- Batch size: 128.
- Warmup is used to prevent divergence: first 1 epoch.
- GPU: Yes.
- **Best result at epoch: 196.**

After pruning the model, the accuracy of the model is very low, so I trained the pruned model for an additional 100+ epochs to raise the score up.

Training the pruned model doesn't increase the memory and time cost as it was simply changing the value of weights and biases.

Training Details – Pruned Model

- The model is trained on the CIFAR100 Dataset.
- 10000 images are used for validation.
- Number of epochs: around 100.
- Initial Learning rate: 0.001.
- Batch size: 128.
- Warmup: first 1 epoch.
- GPU: Yes.
- **Best result at epoch: 100.**

Results

Here is a quick summary of the results:

Criterion / Model	Base Model	Unstructured Pruning Model	**Structured Pruning Model (after training)**
# Parameters	6.4M	6.4M	**4.09M**
File Size (.pth)	25MB	25MB	**16MB**
Top-1 Error	**22.96%**	27.5% (delta 4.54% from base)	25.85% (delta 2.89% from base)
Top-5 Error	**6.19%**	6.61%	6.91%
Runtime Memory	58.66 MB	58.66 MB	**46.98 MB**
Inference Time/Image	15.74 ms	**15.6 ms**	16.7 ms
Total Time CPU	27.395 ms	**26.788 ms**	31.761 ms
Total Time CUDA	12.64 ms	12.64 ms	**10.79 ms**

Unstructured Pruning does not save a lot of memory consumption and time consumption because it has the same number of parameters as the base model, as it uses a masking layer to cancel out the nodes.

Structured Pruning, on the other hand, has 20% less parameters (4.09M) due to nodes being removed completely. However, this model needs to be re-trained so that accuracy score can go up. Due to time constraints, the model can only be trained to an extent.

Others

I have also tried Post-Training Static Quantization; however, I have not been able to reproduce an acceptable model with good accuracy in a short time. This is most likely due to the experimental nature of PyTorch Quantization and the fact the GPU capability is a prototype feature. With more time, Quantization can be a powerful tool to compress models with a more reliable architecture.

Conclusion

In conclusion, these two above compression methods are the easiest to perform on and they can reduce most computation as well as maintain performance if used correctly. This field of model compression is still relatively new and with the integration into PyTorch and TensorFlow will allow these methods to reach more developers.