Report on CIFAR-10 Dataset Image Classification

Yingqi Bian (Team Eternal Core)

Introduction

ResNet, proposed by (He et al. 2015), is a widely used as the backbone component in many of the models in computer vision, such as image classification, instance segmentation and etc. In this report, I will be proposing an improved structure of ResNet (under a limitation of 5 million parameters) for image classification which is trained based on CIFAR-10 dataset. This new structured obtained an accuracy of 84.903% on the benchmark dataset. The code and relevant files can be found in this link https://github.com/yuitoryu/cifar_proj. (Some sentences of this report is written under assistance of ChatGPT.)

Design of Model

My proposed model is calledRes3NetK533. The model is altered from ResNet18 which has about 1.17 million parameters. This allows me to make easy manipulation on the number of parameters. The structure of the model is presented as following.

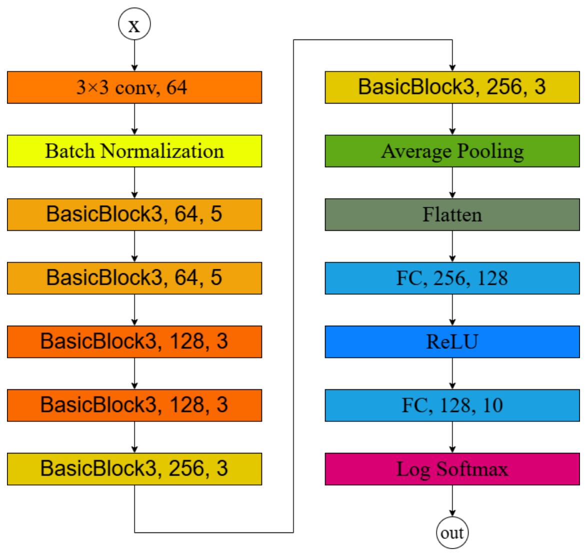

Figure 1: Structure of Res3NetK533

The structure of the sub-module BasicBlock3 is presented as following.

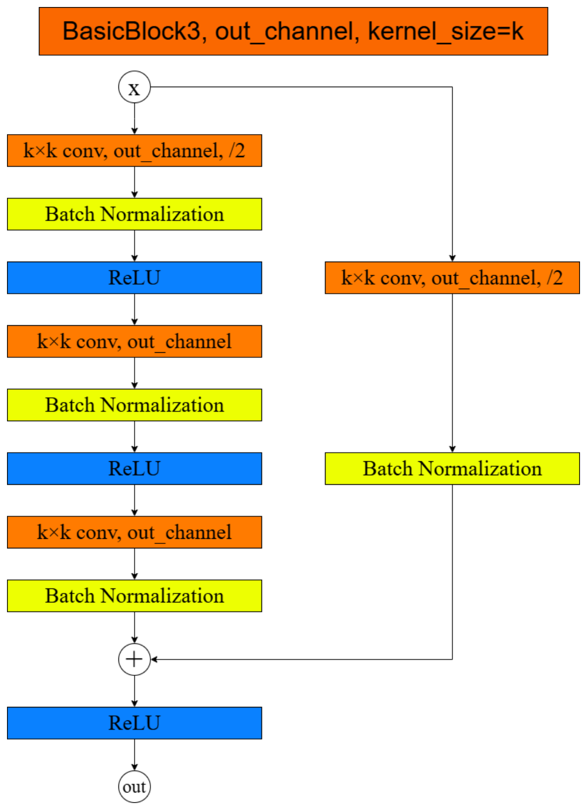

Figure 2: Structure of BasicBlock3

The first major change is that I have removed the 4^{th} residual layer from the original ResNet18. This is completely done for reducing the number of parameters since most of the parameters come from here. Though the most powerful layer might be lost, this left me with a large space with modifying other part of the model. To somehow compensate for the loss, I first added a second linear layer at the very end of the model.

The second change on the original ResNet18 is the residual block. Instead of 2 convolution layers, I increased to 3 convolution layers to improve the ability of feature study of each block.

The third major change is done by altering the size of kernel on the first 2 residual blocks. I hope that a larger size of kernel can help each pixel to gather more local information. Due to the limitation of 5 million parameters, I implemented the larger size kernel in the first 2 residual blocks.

During construction of models, I have also experimented with other structures. For instance, I tried to connect the output of residual layer to Transformer encoders. Although, the training and validation accuracy looks promising, It obtained an accuracy of 80.446%, a score even lower than the basic reduced ResNet18. This was quite disappointing and out of my expectation. One explanation could be that Transformer is hungry on training data and will present problem on overfitting on smaller dataset (Sun et al. 2019).

Training setting

The training is performed on my laptop. The spec is listed by following:

- CPU: i9-14900HX
- GPU: NVIDIA GeForce RTX 4070 Laptop GPU
- Memory: 32GB

A good model must come with good setting on training methodology. Several actions were done for better training result. In this project, the PyTorch framework is used.

Since, out training data are 32×32 images, there should be a lack of features among training data compared to higher resolution images. Thus, data augmentation has to be done. The following actions were taken:

- Random cropping with padding
- Random horizontal flipping
- Brightness, contrast, and saturation adjustment
- Random erasing

Plus, a normalization of $mean = (0.4914, 0.4822, 0.4465)$ and $std = (0.247, 0.243, 0.261)$ (typical setting for CIFAR-10 dataset) has been done to prevent gradient exploding and convergence issue.

The number of epochs were set to be 200 since it is a common setting for ResNet18. I picked SGD as the optimizer since it performs better in task of computer vision than ADAM. I set the learning rate decaying from 0.1 to 10^{-6} and the momentum to be 0.9. The decay is done by Cosine Annealing Learning Rate Scheduler. For stability, I set the first 5 epochs to be warm-up epoch where the learning rate increase from 0.01 to 0.1 linearly.

The loss criterion is NLLLoss since the output of Res3NetK533 is logits.

The training data loader has the following parameters:

- batch_size = 128
- num_workers=12 (enable 12 workers to load data at the same time for faster speed)
- prefetch_factor=4 (preload 4 batches per worker to keep the GPU busy)
- persistent_workers=True (keep all workers alive when changing epoch. Significant performance improving when num_worker is large)
- pin_memory=True (enables faster CPU $\rightarrow$ GPU data transfer by allocating page-locked (pinned) memory)

Result

The following charts are plotted from the training process of Res3NetK533.

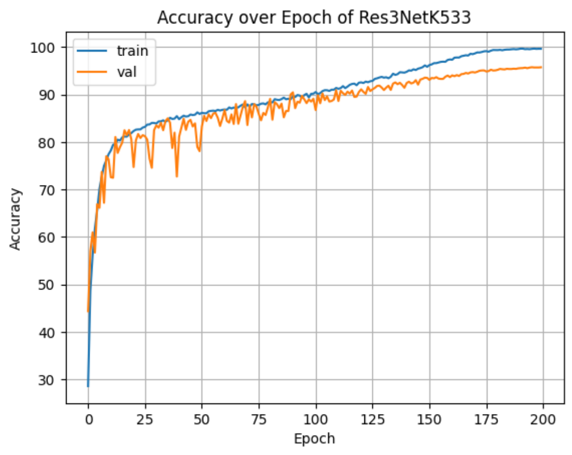

Figure 3: Accuracy over epochs

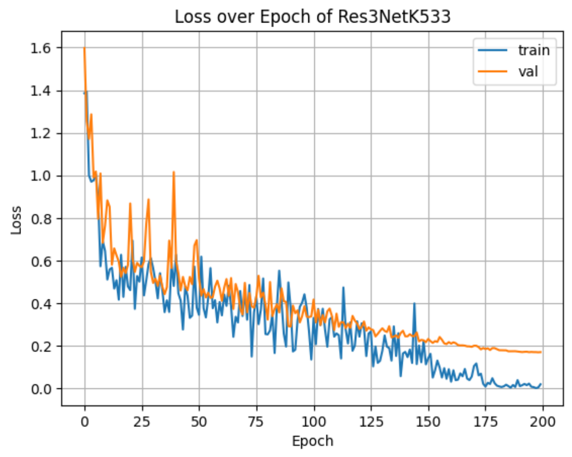

Figure 4: Loss over epochs

The training result is satisfying. Neither underfitting nor overfitting is observed. On the side of Kaggle competition, this model obtained a public score of 85.175% and a public score of 84.903%. The number of parameters is 4752586.

Summary

To sum up, I personally am satisfying with the result of 84.903% with my model Res3NetK533. Throughout the project, I found that speeding up data transfer between GPU and CPU is very important in speed of training. Plus, I have also learned that Transformer is very hungry for data and I should only be considering using this architecture when handing a larger size of dataset.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

Sun, C.; Qiu, X.; Xu, Y.; and Huang, X. 2019. How to Fine-Tune BERT for Text Classification? *CoRR*, abs/1905.05583.