Deep Learning Mini-Project: ResNet on CIFAR-10

Ming-Han Huang
mh6069@nyu.edu
Github Repo: https://github.com/hannnnk1231/ResNet-on-CIFAR-10

Abstract

This project aims getting the best accuracy with a modified ResNet on CIFAR-10, under the constraint that the model has no more than 5 million parameters. In my experiments, I played with different ResNet architectures, optimizers, regularizers, data augmentation strategies and learning rates. In the end, it came out with **94.83%** accuracy.

Methodology

Data Preparation

The dataset for the project is CIFAR-10, which consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. [1]

For the experiments below, I used 90% of the training images to train the model, the remaining 10% of the training images for validation, and tested with the test images. In the final model, I used all training images for training, and tested with the test images.

Model Architecture

In the very beginning, I experimented with different ResNet architectures. Due to the 5M parameter constraint, I first removed the 4-th layer from the basic ResNet-18 architecture, which made it ResNet-14 with **2.7M** parameters. Next, I tried to use as more as parameters I can. I added an additional residual block to each layer, that is, there are 3 residual blocks in each layer. This built a ResNet-20 with **4.3M** parameters.

In this section, I used the basic SGD without any regularization as the optimizer, and trained without any data augmentation for 80 epochs. As the result, the ResNet-14 gave **72.09%** test accuracy and the ResNet-20 gave **68.06%** test accuracy. From Figure 1, we can see that ResNet-20 is not stable after 40 epochs. The model is too complex for the task.

Also, the model suffered from over-fitting. From Figure 2, we can see that the validation accuracy/loss has already stop growing after 50 epochs, but the training accuracy/loss is continue growing. The following section will try to deal with this problem.

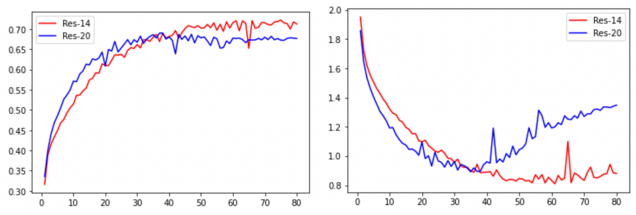

Figure 1: ResNet-14 vs ResNet-20, validation accuracy(left) and validation loss(right)

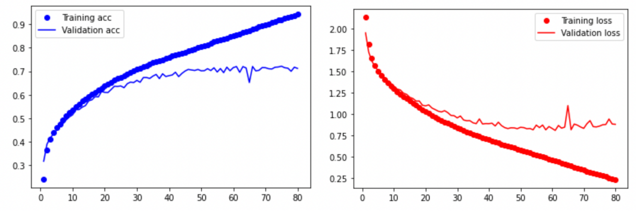

Figure 2: Training result for ResNet-14, validation accuracy(left) and validation loss(right)

Data Augmentation

Data augmentation is the easiest weapon against the over-fitting which can also boost the accuracy. There are few data augmentation strategies:

1. Rotation
2. Horizontal/Vertical Flip
3. Corp

In HW2, we utilized all three strategies for data augmentation (excludes the vertical flip). However, I noticed that most online resources are not using the rotation in their data augmentation part. So, I run an experiment on using the rotation or not. As for the result, using rotation gave **86.25%** test accuracy and not using rotation gave **87.37%** test accuracy.

Comparing Figure 2 with Figure 3, we have improved the over-fitting a lot, and also gave the accuracy a boost. However, there is still some spaces for the model to improve.

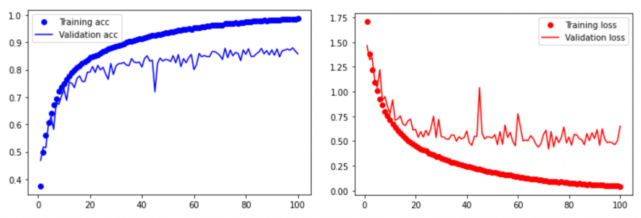

Figure 3: Training result for ResNet-14 + data augmentation (without rotation), validation accuracy(left) and validation loss(right)

Optimizer and Regularization

Optimizer plays an important role in the training process. Choosing a good optimizer for our model can make a big impact. In the previous sections, I choose to use SGD since it's the basic optimizer. However, we can make it stronger by adding the "momentum" into it (SGD-M). Next, I will compare it with the most popular optimizer "Adam" and "AdamW" (Adam with decoupled weight decay). Also, the regularization is a good weapon fighting against the overfitting. To do it, I added the L2-norm to SGD-M by adding the weight_decay parameter to SGD.

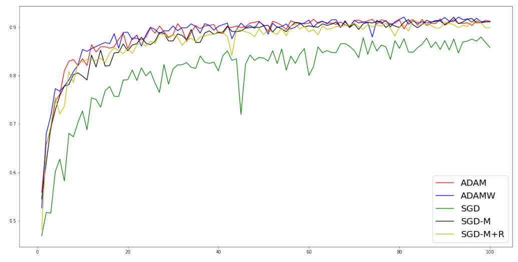

Figure 4: Validation accuracy for different optimizers after 100 epochs.

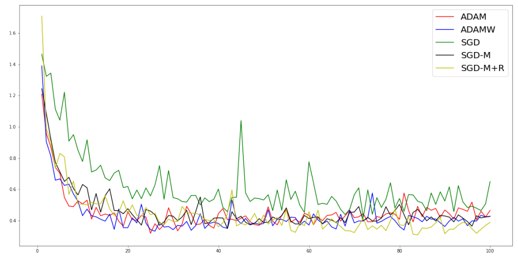

Figure 5: Validation loss for different optimizers after 100 epochs.

From Figure 4 and Figure 5, we can find out that Adam converges faster than the basic SGD, however, adding the momentum to SGD makes SGD can converge as fast as Adam, and gives similar accuracy. In the end, the SGD with momentum and L2-norm gives the lowest loss and the best test accuracy **90.87%**.

SGD	Adam	SGD-M	AdamW	SGD-M+R
87.33	90.03	89.98	89.77	90.87

Table 1: Test accuracy (%) for each optimizer.

Learning Rate

In the last section, I did some experiments on the learning rate. A large learning rate can make the model converge very fast, however, it can cause the model to converge too quickly to a sub-optimal solution. Choosing the right learning rate is also an important factor for the project. The public recommendation learning rate for SGD is 0.01, however, let the result talk.

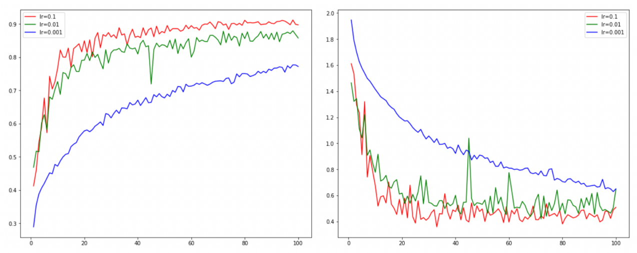

Figure 6: Validation accuracy(left) and validation loss(right) for different learning rate after 100 epochs.

From Figure 6, for learning rate 0.01 and 0.1, it converges very fast. However, it stuck at loss=0.4 to 0.6, which might be the sub-optimal solution. For learning rate 0.001, it converges very slow (and even not reaches the optimal solution after 100 epochs), however, it converges very smooth. To take the benefits from both sides, we can introduce "scheduler" to the project, which will start from a large learning rate, and decrease it by every epoch until 0.

Result

From the above experiment, for the training data, I will apply data augmentation with RandomCorp and RandomHorizontalFlip. I choose to use ResNet-14 with 2.7M parameters in Figure 7 as my final model. For optimizer, I choose to use SGD with momentum and L2-norm, with scheduled learning from 0.1 to 0 after 200 epochs. The final test accuracy is **94.83%**.

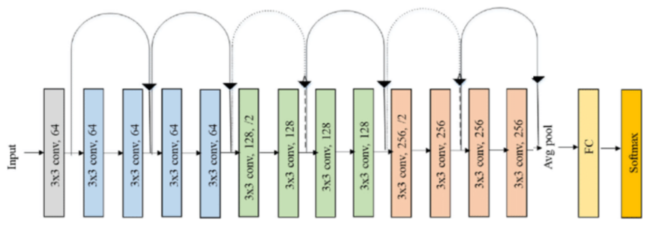

Figure 7: ResNet-14 architecture.

Reference

https://github.com/kuangliu/pytorch-cifar