ResNet on CIFAR-10

Atharva Bhagwat (acb9244), Kostis Paschalakis (kp2405), Sayantan Mukhopadhyay (sm9752)

Link to the repository
Tandon School of Engineering, New York University

Overview

This paper explores a classification problem on the CIFAR-10 dataset using a modified residual network (ResNet) architecture. Moreover the goal is to reduce the size of a typical ResNet (He et al. 2015) to less than 5 million parameters while keeping the accuracy high.

ResNet architecture is any convectional network with skipped connections. The CIFAR-10 dataset consists of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

In our effort to decrease the complexity of the network, and thus the number of parameters we noticed, that the number of layers and the block size in each layer mostly affect the number of parameters. We experimented with two different setups as we will discuss bellow in the methodology section. However, we found that 3 layers with 4, 4, and 3 blocks for each layer seemed to work the best.

Furthermore, we experimented with more typical techniques in order to ensure better training performance and thus a good final accuracy. This included: Data augmentations using normalization, random horizontal flipping, and random crop, as well as selecting the appropriate parameters for the size of train set, validation set, test set, learning rate, batch size, optimizer, weight decay, scheduler etc.

Finally, we ended up with a ResNet network with $4,697,162$ parameters and a final accuracy of 91%.

Methodology

The typical architecture of ResNet is shown in the following figures. Figure 1 shows the typical ResNet architecture. While Figure 2 shows what ResNet looks like with all the layers and blocks.

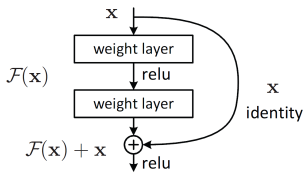

Figure 1: ResNet typical architecture

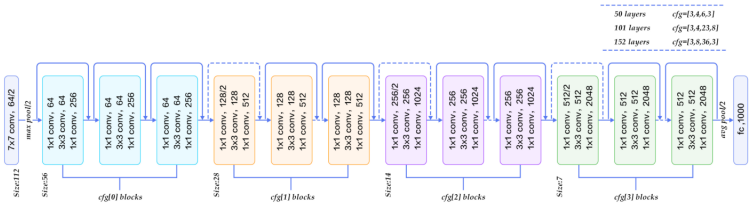

Figure 2: ResNet50 architecture

The first step was to deal with the high number of parameters in a ResNet network. ResNet-18 has around 11 million parameters. Our first intuition which proved to be correct was that we should decrease the number of layers as this would significantly drop the parameters number. In addition, we tried playing around with the blocks in each residual layer. Firstly, we brought down the number of layers to 3. With a 3 layer network, intuitively we designed the number of blocks to decrease with network depth. We also tested the model with 4 layers and reduced number of blocks.

As a result, we ended up with 2 different architectural setups which we decided to test. By test we mean training the network and then evaluating, all with the CIFAR-10 dataset.

Setups:

- Architecture 1 (kuangliu 2021): ResNet with 4 layers with 1 block each: 4,903,242 parameters, acc 89%.
- Architecture 2: ResNet with 3 layers with 4, 4, and 3 blocks for each layer: 4,697,162 parameters 91%.

At this point since training is involved, there are other parameters that are important. We explored different combination of parameters when training, and we concluded that the following seemed to work best:

- train-validation-test size: 45000, 5000, 10000
- Batch size: 128
- Learning rate: 0.01
- Optimizer: SGD with momentum of 0.9 and weight decay of 0.0005

Architecture	Test Accuracy(%)	Parameters
Architecture 1 (4 layers)	89	4,903,242
Architecture 2 (3 layers)	91	4,697,162

Table 1: Test accuracy with different optimizers

Results

As we mentioned above we first tested the two architectures mentioned to decide which one is the best. Their loss graph as well as their final accuracy can be seen bellow:

Train-Validation Loss plot for architecture 1:

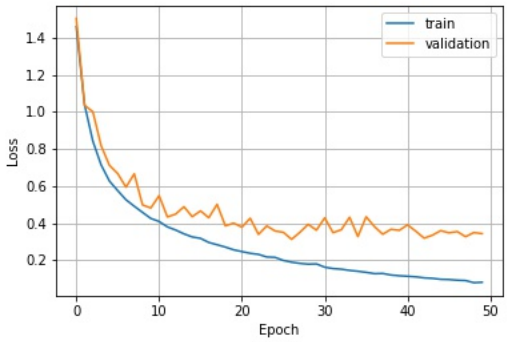

Figure 3: Train-Validation Loss for 4 layer ResNet

Train-Validation Loss plot for architecture 2:

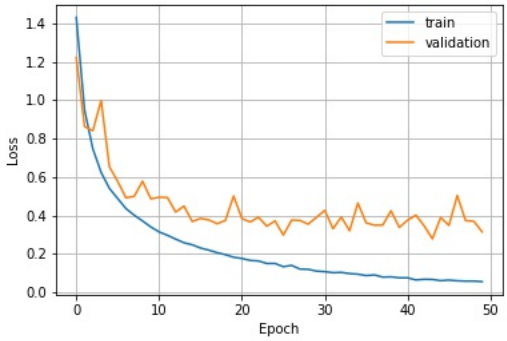

Figure 4: Train-Validation Loss for 3 layer ResNet

Based on these results we selected Architecture 2. ResNet with 3 layers with 4, 4, and 3 blocks for each layer. We then moved into the next step which was to try out different parameters such as learning rate, optimizer etc. We are including a graph of the different optimizers and their final accuracy with Architecture 2. As we can see SGD proved to be the best. We followed a similar approach with all other parameters when training in order to ensure the best performance.

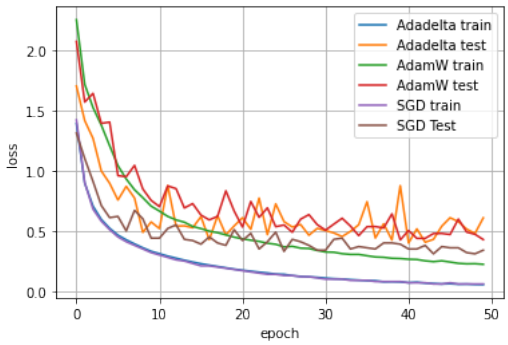

Figure 5: Testing different optimizers

Optimizer	Test Accuracy(%)
Adadelta (Zeiler 2012)	85
AdamW (Loshchilov and Hutter 2017)	87
SGD	91

Table 2: Architectures with Test accuracy

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition.

kuangliu. 2021. Train CIFAR10 with PyTorch. https://github.com/kuangliu/pytorch-cifar.

Loshchilov, I.; and Hutter, F. 2017. Fixing Weight Decay Regularization in Adam. https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html.

Zeiler, M. D. 2012. ADADELTA: An Adaptive Learning Rate Method. https://pytorch.org/docs/stable/generated/torch.optim.Adadelta.html.