Modifications on Resnet-18 Model for CIFAR-10

Siddharth Garg &Arsalan Mosenia
Department of Electrical and Computer Engineering
New York University
sg175@nyu.edu &am12546@nyu.edu

Xueying He
Computer Engineering
xh2311@nyu.edu

Xiao Shen
Computer Engineering
xs2156@nyu.edu

Yuan Tian
Computer Engineering
yt2093@nyu.edu

Abstract

Residual networks (ResNets) have recently been a popular topic on challenging image classification tasks. We introduce a ResNet model that is the modified version of Resnet-18 and achieves better test accuracy within the limit of the number of hyperparameters. Experimental results of implementing our design of ResNet architecture on CIFAR-10 dataset show that the test accuracy of our improved version is 90.86%, and the number of hyperparameters of the model is 4,902,186 which is within the limit of 5M on the number of hyperparameters[1].

1 Introduction

ResNets (or Residual Networks) are one of the most commonly used models for image classification tasks. A ResNet model is constructed by an input layer, a fully connected layer, and an output layer with several residual layers containing a certain amount of residual blocks in between. The most important modification on ResNets compared to normal convolutional neural networks is the "Skip Connections" (or identity mapping) which is implemented on each residual block[2].

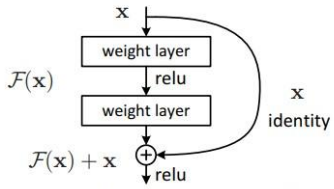

Figure 2. Residual learning: a building block.

Figure 1. A residual block

ResNet-18 is a typical example of ResNet architectures for image classification tasks. Its architecture is shown below. ResNet18 consists of 4 residual layers and each layer contains 2 residual blocks. Each residual block contains 2 convolutional layers with the convolutional layer of kernel size 3. The number of channels of the first residual layer starts from 64 and doubles for later layers, i.e. $C_i = 2C_{i-1}$ for $i = \{2, 3, 4\}$.

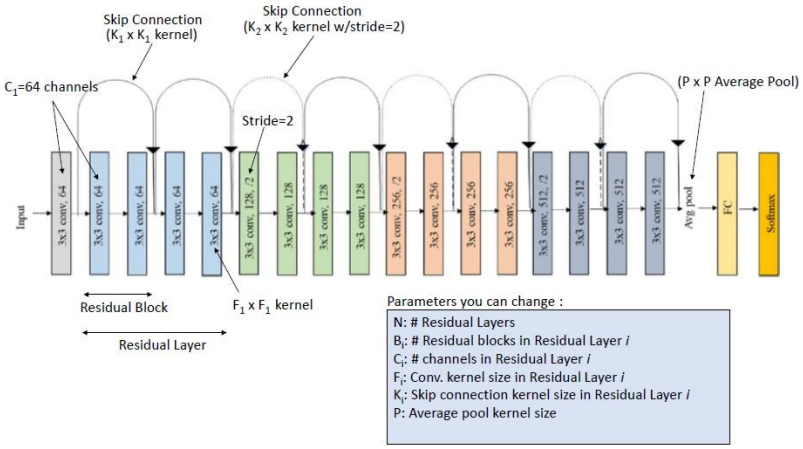

Figure 2. Architecture of ResNet18

In this paper, we will introduce our own ResNet model for CIFAR-10 image classification. We have attempted different designs with modifications on six hyperparameters and tested different combinations of those hyperparameters to obtain the model whose number of hyperparameters is within the limit of 5M and accuracy of the test image set is maximized. We finally obtained our optimal ResNet model satisfying the requirements on both the number of hyperparameters and the test accuracy.

2 Methodology

2.1 ResNet Architecture Design

We used the Control Variable Method to split six hyperparameters to all of us so that we could discover the role each hyperparameter plays in this ResNet-18 model.

We kept other hyperparameters unchanged, only change N (Residual layer) and B_i (Residual blocks in Residual Layer i) for the very first test . We changed the residual layer and Residual blocks respectively, and tried to make the parameters not exceed 5M. The best test accuarcy we obtained is by setting the number of residual layers to 4, and the number of blocks of each layer to 1, and the accuracy we got is 91.78%.

We then built three Resnet classes with different numbers of channels in Residual Layer i and four BasicBlocks with different convolutional layer kernel sizes in Residual Layer i, and detected the number of hyperparameters and highest accuracies of their combinations. As a result, we found that Resnet model with C_1=32, $C_i = 2C_{i-1}$ for $i = \{2, 3, 4\}$ and convolutional kernel size F_i = 3 produced the highest accuracy which is 86.6%.

As for the hyperparameters K_i (skip connection kernel size in Residual Layer i) and P (Average pool kernel size), we tried different combinations with other hyper-parameters to make sure that our ResNet model has less than 5M trainable parameters. Finally, we decide to use $K_i = 1$ and P = 4 for our model and achieved the test accuracy of more than 90%.

We record the model accuracy with different combinations of hyper-parameters in Table 1. We keep the same training process, where we use around 50 epochs. (Final training with 80 epochs)

Table 1: Model Accuracy with different hyperparameter combination

Num	Model Accuracy	#Parameters	#Residual layers	#Residual blocks	#Channels	Conv. kernel	Skip connection kernel	Average pool kernel
1	84.8%	1.9M	4	[2,2,2,2]	[16,32,64, 128]	5	1	4
2	86.6%	2.79M	4	[2,2,2,2]	[32,64,12 8,256]	3	1	4
3	91.78%	4.9M	4	[1,1,1,1]	[64,128,2 56,512]	3	1	4
4	89.10%	4.9M	4	[1,1,1,1]	[64,128,2 56,512]	3	1	3
5	88.43%	5.4M	4	[2,2,2,2]	[64,128,2 56,512]	3	2	4
6	90.70%	4.9M	4	[4,6,4,3]	[32,64,12 8,256]	3	1	4
7	90.93%	4.99M	4	[5,7,4,3]	[32,64,12 8,256]	3	1	4

Based on our previous experiments, we found that when the number of channels of each layer were [32, 64, 128, 256] respectively and the kernel size of the convolutional layer of each block was 3, we obtained higher accuracy. Therefore, our main direction is to modify other hyperparameters to increase the number of hyperparameters and improve the performance of the model.

We finally decided to change other parameters based on 4 Residual layers. The reason is that 5 layers would exceed 5M parameters easily. In addition, when we tried to decrease the layers, like 3 layers, the parameters would be greatly reduced, which lowered the test accuracy, even though we tried to increase other parameters, the accuracy was still low. As a conclusion, 4 Residual layers is a relatively optimal choice.

2.2 Technique selection in training process

In order to increase the amount of data to improve the performance of our model, we would use data augmentation techniques to generate new data points from the existing data. We used RandomHorizontalFlip(), RandomCrop() with the desired output size 32, padding with 4 on each border of the image and normalization with mean = [0.4914, 0.4822, 0.4465], standard deviation = [0.2023, 0.1994, 0.2010]. Random crop method would crop images during the training process, which means we create a random subset of an original image to represent the objects of interest in the various situations. In addition, the reason we use RandomHorizontal Flip method is that some objects might face different directions so that horizontal flip would complement this sort of data to generalize our model.

We choose a batch size of 128 instead of 64 since the dataset is relatively large and the larger batch size would speed up the training process. We also decided to use Adam as our optimizer. We

have attempted to use SGD (Stochastic Gradient Descent) as our optimizer, however, we found that the test accuracy produced by SGD is much lower than the one returned by Adam with the same setting of learning rate and model hyperparameters. Figures 3-6 show the comparison of both accuracies and losses of two optimizers. It is obvious that the highest accuracy produced by SGD does not even pass 80% and the training loss is also greater than that generated by Adam.

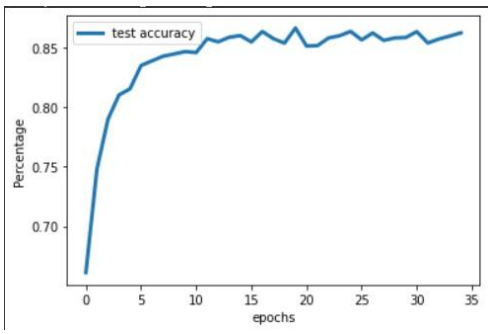

Figure 3. Adam test accuracy

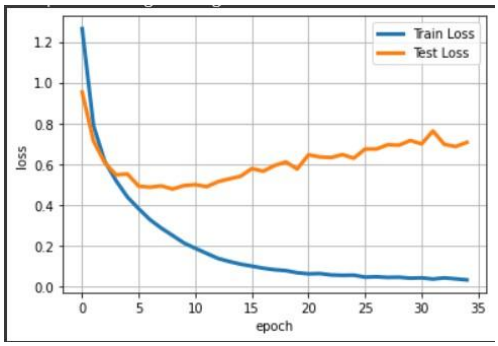

Figure 4. Adam training and testing losses

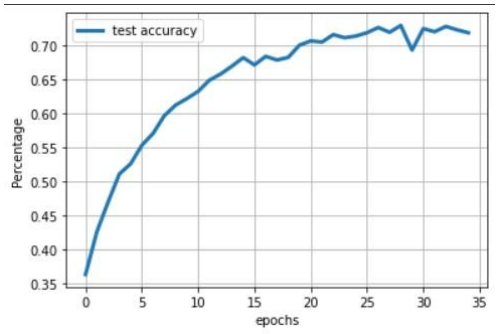

Figure 5. SGD test accuracy

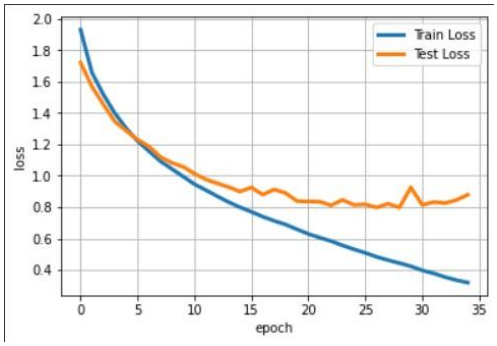

Figure 6. SGD training and testing losses

As for the number of epochs, we initially used 120 epochs to train the model for higher accuracy. However, we found that the model overfitted at 80 epochs, thus we decided to stop early at 80 epochs.

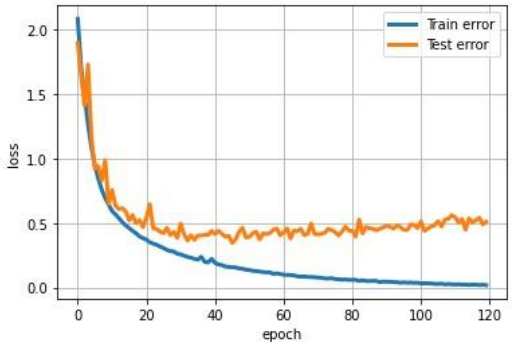

Figure 7. model overfitting in 120 epochs

3 Results

3.1 Final ResNet Architecture

For our final ResNet model, we decided to use 4 Residual Layers, [4, 6, 4, 3] residual blocks in Residual layer i, [32, 64, 128, 256] channels in residual layer i, 3 convolution kernel size in all residual layers, 1 skip connection kernel size in residual layer, 4 average pool kernel size. (The same as the 6 combinations in Table 1). The total number of trainable parameters is 4,902,186 which is less than 5M.

3.2 Details of how the model was trained

3.2.1 data load and preprocessing

Firstly, we load CIFAR-10 image data and split it as training data (50000) and testing data (10000). We use a data loader for them to provide an iterable batch over the given dataset, where we set batch size as 128 and num_workers as 2 to speed up the training process. In addition, we use RandomCrop and RandomHorizontalFlip techniques to augment data for better model performance. Then we performed the normalization of inputs, which will help to change the range and scale of data.

3.2.2 Optimizer and training

Secondly, we decide to use cross entropy as a loss function and Adam as an optimizer. We initially set up lr = 0.1 and use the learning rate decay method (cosine annealing learning rate) which allows dynamic learning rate reduction based on some validation measurements. Then, we started to train our ResNet model with 80 epochs by using gradient descent algorithm.

3.3 Result Display

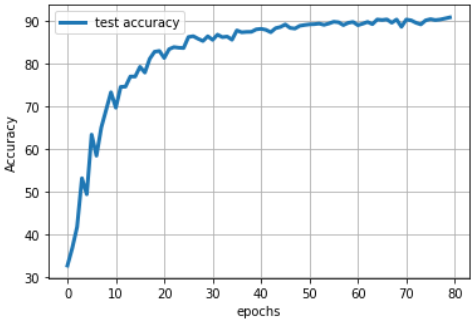

Figure 8. accuracy versus epochs

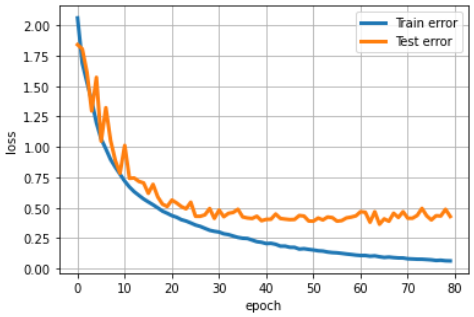

Figure 9. losses versus epochs

The final test accuracy we obtained is 90.86%, and the testing loss is 0.42.

4 Conclusion

By decreasing the number of channels of each residual layer to the half of the base model and increasing numbers of residual blocks to [4, 6, 4, 3], we finally obtained the highest test accuracy 90.86% among all of the models we attempted and reached the number of hyperparameters 4902186 which is within the 5M constraint. Hence, we could conclude that while keeping other hyperparameters unchanged and just changing the number of channels and number of blocks of each layer, the closer we are to the 5M constraint, the higher the test accuracy we could get.

5 References

[1] B. Li and Y. He, "An Improved ResNet Based on the Adjustable Shortcut Connections," in IEEE Access, vol. 6, pp. 18967-18974, 2018, doi: 10.1109/ACCESS.2018.2814605.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385

[3] DeepLearnPhysics (2021) pytorch-resnet-example[source code]

https://github.com/DeepLearnPhysics/pytorch-resnet-example

[4] kuangliu (2021) pytorch-cifar[source code]

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

6 Github repository

Here is our github repository for our modified ResNet model: **pytorch_cifar_resnet**.