2022 Fall Deep Learning Mini-Project
A Modified Residual Network on CIFAR-10 Dataset

Yichen Guo,[1] **Tao Liang,** [2] **Zhaorui Ma,** [3]

[1] Center for Urban Science Progress. New York University Tandon School of Engineering, United States
[2] Department of Civil Engineering. New York University Tandon School of Engineering, United States
[3] Department of Electrical and Computer Engineering. New York University Tandon School of Engineering, United States
GitHub link: https://github.com/liangtao1216/ResNetminiproject.git

Abstract

Deep neural networks have been proven effective on the field of image processing. Scholars have achieved significant milestones on the use of deep residual neural networks (ResNets) in training standard CIFAR-10 dataset. In this paper, we presented a modified residual learning framework that achieves the optimal test accuracy on the CIFAR-10 dataset under the constraint of parameter count. Our best model achieved an accuracy of 0.95 with 100 training epochs.

1. Methods

We started our model framework exploration based on previous literature (He et al. 2016) and code documentation (kuangliu 2021). By studying previous works, we further understand the dataset and factors that would affect accuracy: optimizer, regularizer, and other parameters such as learning rate and batch size. Below is a list of elements we identified to experiment with:

1. Data augmentation method
2. Hidden Layer Size
3. Combination of blocks
4. Optimizer (SGD, Adam)
5. Learning rate (fixed learning rate, adaptive learning rate)

1.1 Model Design

The higher level of the model design was based on the parameter constraint of 5 million. We modified the combination of blocks and hidden layer to satisfy the parameter constraint. First, we chose two network structures for comparison, one modify based on Resnet 18, another one used bottleneck structure, see Figure 1. After the general framework was selected, we continued to compare the relative influence of the hyperparameters selection such as optimizer, learning rate.

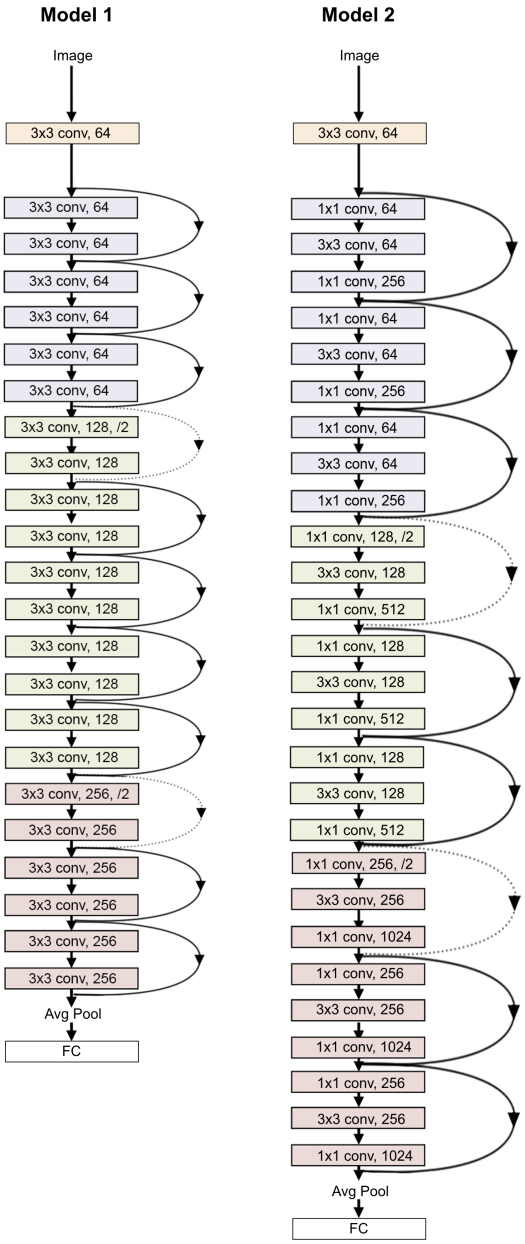

Figure 1: Network architectures for CIFAR 10. **Left:** Model 1. **Right:** Model 2. The dotted shortcuts denote the change of the dimension. **Table 1** shows more details and other variants

Layer Name	Output Size	Model 1	Model 2
conv1	32×32	3 x 3, 64, stride 1	
conv2_x	32×32	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	16×16	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 5$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 3$
conv4_x	8×8	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 3$
	1×1	Avg pool, FC, Softmax	
Params		4,918,602	4,914,250

Table 1: Architectures for CIFAR 10

1.1.1 CIFAR-10 Dataset The CIFAR-10 dataset has 10 classes which are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Each class contains 6000 32x32 color images. The CIFAR-10 dataset has 60000 images which can be separate into 50000 training images and 10000 test images. There are 5 training batches and 1 test batch and each batch contains 10000 images. The test batch selects exactly 1000 images from each class randomly. The training batches contain the remaining 50000 images in random order, each training batch may not have exactly 1000 images from each class. But among them, the training batches have exactly 5000 images from each class(Krizhevsky 2010). In order to prevent overfitting problem, we used data augmentation method, crop and flip the image randomly, to process the data.

1.2 Model Training

The models were trained after careful consideration of choices of parameters, and we tried to keep the number of parameters as close as 5 million. Finally, the models' performance were compared within 100 epochs.

1.2.1 N, Number of Residual Layers The number of residual layers is directly correlated with the depth of the network. We calculated the number of parameters of ResNet18 designed by kuangliu (kuangliu 2021), and turned out to have 11 million parameters which was a lot more than our constraint. So we decided to reduce the number of layers based on the ResNet18 structure since CIFAR-10 is a relatively small dataset. So for Model 1, we had 24-layers. Since bottleneck structure allowed us use less parameters to construct deeper network. So for Model 2, we had 29 layers.

1.2.2 C_i, Number of Channels in the i_{th} Layer The number of channels greatly affected the number of parameters. For each residual layer, the number of parameter would at least doubled. Thus we took careful consideration of our initial choice of our channel sizes. We targeted our initial channel size at 64.

1.2.3 B_i, Residual Blocks in i_{th} Layer Larger filter size would be computationally cheaper. However, it would also loss some information of the input.Since we were working on the CIFAR-10 dataset, smaller filter size may keep more information. So we decided to follow the modified ResNet18 structure from (kuangliu 2021), with initial 3×3, stripe 1 convolution layer and a batch normalization layer to process the input, kept the feature map as 32×32 as the dimension of CIFAR 10. For model 1, we used two 3×3 filters followed by a batch normalization layer respectively. For model 2, we used bottleneck structure, a 1×1 filter, a 3×3 filter, and a 1×1 filter.

1.2.4 K_i, Kernel Size in the i_{th} Skip Connection Since ResNet calculates $H(x) = F(x) - x$, rather than calculate $F(x)$. So for each ResNet layer, we had to send the input to combine with the layer output. However, the feature map of output and input might not be the same dimension. So we used a 1×1 filter with adaptive stripe to process the input at each residual block.

1.2.5 P, Pool Size in the Average Pool Layer Since we used both BottleNeck and BasicBlock, we used adaptive average pooling rather than a fixed average pooling size. In other words, the average pooling size was calculated based on the input and output dimensionality.

1.2.6 Other Parameters We evaluated the performance of Adam and SGD based on the accuracy on the test data. For learning rate, we compared the model performance of fixed learning rate and decaying learning rate, Cosine Annealing (see Figure 2).

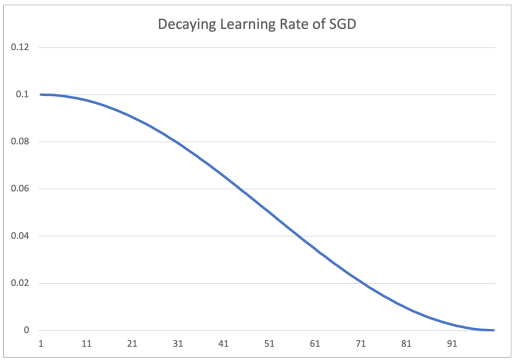

Figure 2: Cosine Annealing Decaying Learning Rate

1.3 Architecture Summary

Table 1 shows a summary of the final architectures and parameters. Model 1 used the Basic Resnet Block while Model 2 integrated the BottleNeck structure. A visualization of the model architecture is shown in Figure 1.

2. Results

Table 2 shows an overview of the final model performances within 100 epochs on the test set.

2.1 Architecture Comparison

Figure 3 shows a comparison between the test and train loss for Model 1-1, 1-2 and Model 2-1, 2-2, we can see that Model 1 sets converged faster than Model 2 sets and have better final performance.

Figure 3: Train and Test Loss of Model 1 and Model 2. Light lines denoting test loss, and bold lines denoting train loss.

2.2 Optimize Method Comparison

Figure 4 shows a summary of the test and train loss for Model 1-1 through Model 1-4, we can see that the models used Adam converged faster than used SGD, and the models used decaying learning rate technique have better performance.

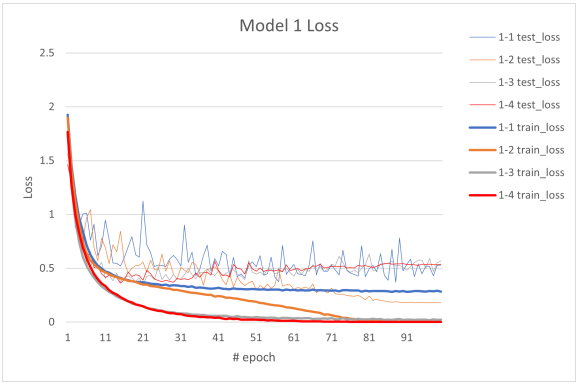

Figure 4: Train and Test Loss of Model 1.Light lines denoting test loss, and bold lines denoting train loss.

Model	Layers	Optimizer	Accuracy
Model 1-1	24	SGD, lr=0.1	87.91%
Model 1-2	24	SGD, Decaying lr	95.43%
Model 1-3	24	Adam, lr=0.01	92.06%
Model 1-4	24	Adam, Decaying lr	93.07%
Model 2-1	29	SGD, lr=0.1	84.90%
Model 2-2	29	SGD, Decaying lr	94.19%
Model 2-3	29	Adam, lr=0.01	90.56%
Model 2-4	29	Adam, Decaying lr	91.42%

Table 2: Model performance

3. Conclusion

We trained two models with different structures. A residual layer was removed from both Model 1 and Model 2 compared to the structure proposed by kuangliu. Model 1 adapted the basic structure of ResNet 18 and ResNet 34 and model 2 used the BottleNeck structure (kuangliu 2021). By comparing the two models, Model 1 performed better than Model2.

For optimizer, we concluded that the SGD optimizer performed better than the Adam, and the adaptive learning rate was better than a fixed learning rate in out training.

Our best model turned out to be Model 1-2 with a total of 4,918,602 parametrs, which used a BasicNet structure with decaying learning rate and SGD optimizer. In the future, we may take runtime into consideration and try to converge to the best result in less epochs.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Krizhevsky, A. 2010. CIFAR-10 Dataset. https://www.cs.toronto.edu/~kriz/cifar.html.

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