A Modified ResNet with Pretty High Test Accuracy on CIFAR-10

Jincheng Liang@jl12003, Shaomin Xu@sx2311, Taishan Zhao@tz2516

Abstract

In this paper, we are presenting a modified residual network architecture focusing on achieving the highest test accuracy on the CIFAR-10 image classification dataset, while strictly adhering the constraint of not exceeding 5 million parameters in our model. We implemented 4 modifications to the architecture, including the number of Residual layers, blocks in Residual layer, number of channels in Residual layer, and average pool kernel size. In order to pursue higher accuracy, we also adjusted learn rate and number of epochs in the training process. We trained and evaluated our modified models on the CIFAR-10 dataset, reached a test accuracy of 95.95%, which competes with previous state-of-art models under similar parameter constraints. Our comparison experiment results also demonstrated the relationship between each of our hyperparameter and the test accuracy of the ResNet architecture. The project codebase and trained models can be accessed at https://github.com/JinchengLiang/DLminiproject.

Introduction

Deep learning revolutionized the entire field of computer vision by achieving outstanding test accuracy in image classification tasks. However, an increase in model complexity and scale result in an inevitable increase in the number of trainable parameters. It would be challenging to deploy such models on devices that are resource-insufficient, which makes the design of an efficient deep neural network architecture a growing interest.

Residual Network(He et al. 2015) (ResNet) has been influential ever since its introduction. This architecture remains magnificent in numerous scientific fields for facilitating the training and optimization process of deep networks with its residual connection feature. In this project, our objective is to improve the test accuracy of image classification on CIFAR-10, which is a popular dataset for image classification tasks, using a modified ResNet architecture, while not exceeding the limit of 5 million parameters. We implemented 4 hyperparameters on the archiecture, including the number of Residual layers, blocks in each residual layer, number of channels in each residual layer, and average pool kernel size, then studied their effects on the test accuracy for future improvements. We also made adjustments to the learn rate and number of epochs in the training process to pursue higher accuracy.

After training the ResNet26-B453 architecture with learn rate 0.1 in 600 epochs, we reached the maximum test accuracy of 95.95% when evaluating our model on the CIFAR-10 dataset. The result is fascinating, as the accuracy competes with previous state-of-art models under similar parameter constraints, and demonstrates how each hyperparameter can affect the final performance of the architecture.

Methodology

Reduce Parameters

ResNet18 has 11,173,962 trainable parameters. In order to get a ResNet that has no more than 5 million parameters, we can change some hyperparameters of ResNet18. Hyperparameters can be changed to reduce trainable parameters: (a) N: the number of residual layers, (b) B_i: the number of residual blocks in residual layer i, (c) C_i: the number of channels in residual layer i, (d) F_i: convolution kernel size in residual layer i, (e) K_i: skip connection kernel size in residual layer i, (f) P: Average pool kernel size. In ResNet18, $N = 4$, $B = [2, 2, 2, 2]$, $C = [64, 128, 256, 512]$, $F = [3, 3, 3, 3]$, $K = [1, 1, 1, 1]$, $P = 4$. The simplest way is to change B to be half, since the half of the number of parameters of RestNet18 is about 5 million. We get data in Table 1 in this way.

Model	B	Params	$\leq 5M$
ResNet10	[1, 1, 1, 1]	4,903,242	✓
ResNet12	[2, 1, 1, 1]	4,977,226	✓
ResNet12-B	[1, 2, 1, 1]	5,198,666	×
ResNet14	[3, 1, 1, 1]	5,051,210	×

Table 1: Total Trainable Parameters

When only B is changed, ResNet10 and ResNet12 succeed, while ResNet12-B and ResNet14 is failed. Therefore, the method that continues to increase B_i will fail if not changing other hyperparameters.

Training Models

As with the basic procedures in model training, we first selected the classic cross-entropy loss function to help us ad-

just model weights during the training process in a classification task with more than two classes. The cross-entropy loss function(Kiprono Elijah Koech 2020) is defined as below:

$$L_ce = -\sum_{i=1}^{n} t_i \log(p_i)$$

Where n is the total number of classes for classification, t_i is the truth label and p_i is the Softmax probability for the i^{th} class. The models weights are manipulated to minimize the cross-entropy loss. The optimizer we selected for the optimization process was stochastic gradient descent (SGD) with a initial learning rate of 0.1 and momentum of 0.9. Meanwhile, we utilized a scheduler CosineAnnealingLR to set the learning rate for each parameter group using a cosine annealing schedule. Our data augmentation strategy was to randomly flips images horizontally. It facilitated the generalization process without modifying the labels or classes, reducing over-fitting occasions and improving overall accuracy. This technique is extremely helpful when the dataset is relatively small.

After training the models in 200 epochs, the ResNet10 architecture achieved an accuracy of 94.66%, while the ResNet12 architecture achieved an accuracy of 92.12%. Surprisingly, more layers, less accuracy. We will analyse layers more in section Residual Layers and section Channels.

Modify Hyperparameters

Besides the hyperparameters mentioned ahead in section Reduce Parameters that can influence the trainable parameters of a model, there are some parameters relating to training process, such as optimizer, data augmentation strategy, regularizer and choice of learning rate, batch size, epochs. We will modify some of both types of hyperparameters to get a model with better accuracy and to analysis the attributions of the hyperparameters to the accuracy.

Learning Rate. Selecting an appropriate learn rate is often experimental. A learn rate that is too low may significantly slow down the training process, or possibly stuck in a local minimum. On the other hand, a learn rate that is too fast may cause the model to oscillate or diverge, lowering the accuracy even more. Cosine annealing(Loshchilov and Hutter 2017) is a popular technique for adjusting the learning rate during the training of deep neural networks. It is a type of learning rate scheduling that gradually reduces the learning rate to zero over a certain number of iterations or epochs. We set the maximum learning rate of ResNet10 from the initial 0.1 to two different values 0.01 and 0.001 to get test results in Table 2. The model whose learning rate is 0.1 converges slowest and oscillate in the first 125 epochs (Figure 1). As a result of scheduler CosineAnnealingLR to reduce the learning rate, this model becomes steady and arrives at the highest accuracy in the end.

Residual Layers. Increasing the number of residual layers have its own pros and cons. With additional layers, the network is able to learn features that are more complex, leading to a possibly higher accuracy. However, too many residual layers may cause issues such as over-fitting, where the model loses the ability to learn from new data other than the ones it is trained on. We trained and evaluated two more ResNet architectures that have different number of residual layers (N) from ResNet10, meanwhile made sure not to exceed the parameter constraint. Based on the returned result (Table 3), the ResNet26-B453 architecture with three residual layers achieved the highest accuracy of 95.64%. The result seems show that more layers means higher accuracy but stacking many layers with small number of channels doesn't work, like ResNet56.

Model	LR	Acc(%)
ResNet10	0.1	**94.66**
ResNet10/100	0.01	93.45
ResNet10/1000	0.001	90.27

Table 2: Learning Rate

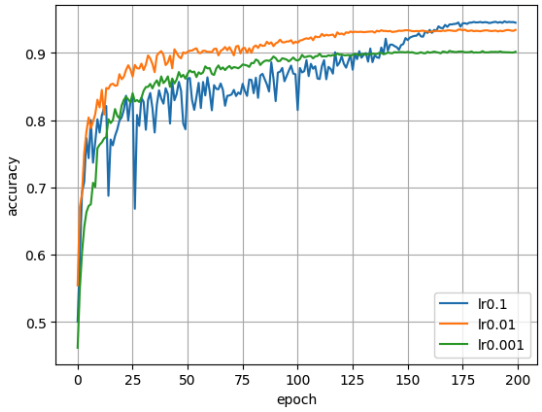

Figure 1: Learning Rate

Model	N	Params	Acc(%)
ResNet10	4	4,903,242	94.66
ResNet26-B453	3	4,992,586	**95.64**
ResNet56	2	4,818,378	94.74

Table 3: # Residual Layers

Residual Blocks. A ResNet architecture may consist multiple residual blocks. Each residual blocks consists one or multiple convolutional layers and a skip connection, which runs through two convolutional layers. The purpose of such design is to prevent the vanishing gradient problem by allowing gradients to flow directly through the network. In general, the selection of an ideal residual block structure involves experimental approaches. After comparing two different residual block combinations under the stated constraints $([4, 5, 3] and [5, 4, 3])$, the difference in the outcome accuracy was hard to tell. We decided to continue the experiment with the more accurate architecture, which is ResNet26-B453.

Channels. Theoretically, increasing the channel size will improve the accuracy of the ResNet architecture, especially

Model	B	Acc(%)
ResNet26-B453	[4, 5, 3]	**95.64**
ResNet26-B543	[5, 4, 3]	95.60

Table 4: # Residual Blocks

when training deeper and larger scaled networks. However, the overall impact also depend on other parameters(Liu et al. 2017). With ResNet26-B453 architecture, we changed its number of channels in residual layer 3 from 256 to 192. Surprisingly, the accuracy only reduced slightly, compared to the drastic drop in the number of trainable parameters, which decreased about one third due to this modification. Therefore, we expected that the performance would be better if we increased the number of trainable parameters by adding the same residual blocks in the residual layer 3. We then obtained a model ResNet30 whose number of trainable parameters is same as ResNet26-B453 by adding two additional residual layers with the same convolutional structure ($B = [4, 5, 5]$). However, the accuracy decreased significantly, seem meaning that big number of channels is important but too many layers with such channels is bad. The reason we guess is that the CIFAR-10 image dataset is much simpler than ImageNet and it doesn't need too many channels to get features.

Model	C	Params	Acc(%)
ResNet26-B453	[64, 128, 256]	4,992,586	**95.64**
ResNet26-B543-C	[63, 128, 192]	3,618,890	95.60
ResNet30	[63, 128, 192]	4,947,530	95.46

Table 5: # Channels

Model	Epochs	Acc(%)
ResNet26-B453	200	95.64
ResNet26-B453-E4	400	95.75
ResNet26-B453-E6	600	**95.95**

Table 6: Epochs

Epochs. Training the model with insufficient epochs may lead to under-fitting and inaccurate results, but too many epochs can also be risky. It may result in over-fitting the model, while being computational expensive. It is mostly recommended to monitor the accuracy of the model throughout the training and validation process. When the test accuracy hardly improves or even starts to fall, it might be a sign for over-fitting. After increasing the number of epochs from 200 to 400, and in the end, 600, we observed a steady trend in the test accuracy curve (Figure 3), where the accuracy reached 95.95% (Table 6) and stayed within the range. More epochs means more training time. The epochs increases double but the accuracy increases so slightly. We believed further training wouldn't be necessary in this situation.

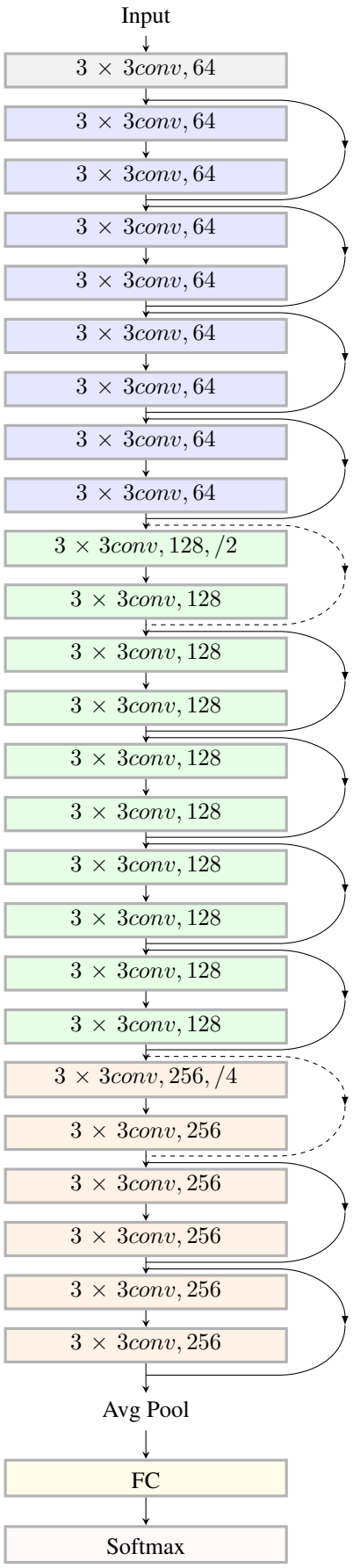

Figure 2: The Architecture of ResNet26-B453-E6

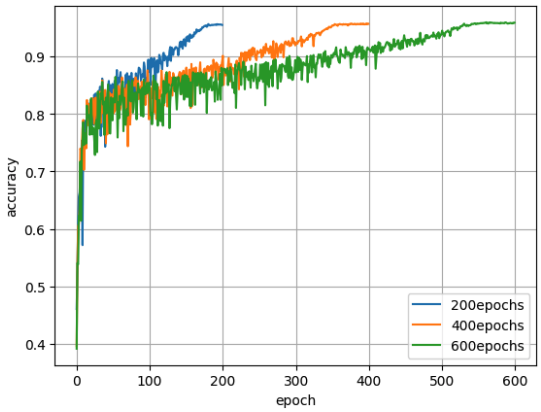

Figure 3: Test Accuracy Development After Different Number of Epochs

Average Pool Kernel Size. A larger average pool kernel size would downsample the input more aggressively and theoretically, leading to a loss of information and possibly lower accuracy. However, the final result also depends on other hyperparameters. After evaluating the ResNet26-B453 architecture with three average pool kernel sizes $(2, 4, 8)$, the model surprisingly achieved its highest test accuracy of 95.88% at a pool size of 8. It demonstrate how import experimental approach is when selecting an ideal value for a hyperparameter.

Model	P	Acc(%)
ResNet26-B453-P2	2	87.12
ResNet26-B453	4	95.64
ResNet26-B453-P8	8	**95.88**

Table 7: Average Pool Kernel Size

Result

After modification and evaluation, the highest test accuracy of **95.95%** was achieved by the ResNet26-B453-E6 architecture on the CIFAR-10 image classification dataset after 600 epochs of training. This model consists a total of **4,992,586** parameters, adhering the constraint. This architecture is demonstrated in Figure 2: the number of residual layers $N = 3$, the number of residual blocks in each residual layer $B = [4, 5, 3]$, the number of channels in each residual layer $C = [64, 128, 256]$, convolution kernel size in each residual layer $F = [3, 3, 3, 3]$, skip connection kernel size in each residual layer $K = [1, 1, 1, 1]$, average pool kernel size $P = 4$.

Conclusion

We constructed a high-performance image classifier using a modified ResNet26 architecture, trained and evaluated out models using the CIFAR-10 dataset. According to our experiment results, under the same constraint, a ResNet architecture with more residual layers, channels, and greater pool size performs better, and the residual block structure as well as learn rate should be carefully obtained through experimental approach. The accuracy may further improve after conducting more experiments and validations on the indicated hyperparameters and more epochs of training, with close monitoring to avoid problems such as over-fitting.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Kiprono Elijah Koech. 2020. Cross-Entropy Loss Function. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e. Accessed: 2023-04-13.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning Efficient Convolutional Networks through Network Slimming. arXiv:1708.06519.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv:1608.03983.