Finely Tuned RestNet-18 on CIFAR-10

ZHANG YU

November 18, 2021

1 Introduction

1.1 CIFAR-10 Introduction

The CIFAR-10 dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.[3] The 10 classes are *plane, car, bird, cat, deer, dog, frog, horse, ship, truck*. Figure 1 include some examples from the training set. By investigating these samples, we can see that some pictures are not so standard and hard to distinguish even by human. The following problems may need attention in data prepossessing.

1. Pictures vary greatly within one class. For example, pic(2,7) and pic(3,7) are both classified as horse, while the former is captured from the side and the latter is captured from the front.

2. Pictures contain more than one object. Examples are like pic(1,1), pic(1,7), and pic(3,3). They contain objects irrelevant to the 10 labels, like building and human.

3. The number of objects are more than one. Pic(2,3) contains two cats.

4. Pictures contain only a part of objects in the corners, like pic(3,1).

Jasper Snoek has reported a test error rate of 15% (without data augmentation), where he used Bayesian hyperparameter optimization to find nice settings of the weight decay and other hyperparameters. Other results include 18% test error without data augmentation and 11% with. [1]

Figure 1: 24 samples from training set of CIFAR-10. Pic(i,j) means the picture in the i-th row and the j-th column.

1.2 Network Architecture

ResNet-18 contains two different kind of residual blocks. As shown in figure 2, the left block is the regular residual block, and the right block is the residual block with 1×1 convolution.[2] With residual blocks, inputs can forward propagate faster through the residual connections. If we want to change the number of channels, we need to introduce an additional 1×1 convolutional layer to transform the input across layers.

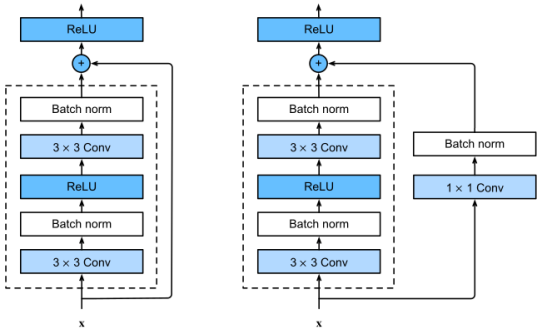

Figure 2: ResNet block with and without 1×1 convolution.[5]

Figure 3 depicts the entire ResNet-18 for CIFAR-

10. The first block called *stem* is a 3×3 instead of 7×7 convolutional layer. We don't perform downsampling, because the image size in CIFAR-10 is 32×32. The resolution decreases while the number of channels increases up until the point where a global average pooling layer aggregates all features.

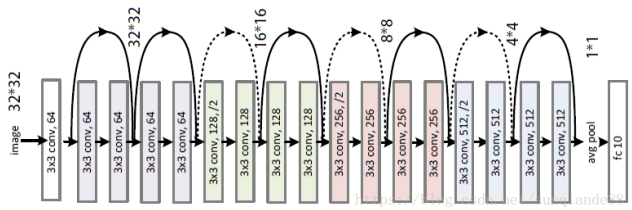

Figure 3: The ResNet-18 architecture for CIFAR-10.[4] The curve with solid line means identity map. The curve with dash line means 1×1 Convolutional layer.

2 Method and Result

In this section, the ResNet-18 is implemented in PyTorch. By finely tunning the hyperparameters, including learning rate, learning rate schedule, weight decay, the model is gradually improving. Further, data augmentation is applied for better model generalization. Initially, the training process does not include learning rate schedule and weight decay, and the data preprocessing only contains random crop, random horizontal flip and mean-std normalization.

2.1 Learning Rate

In machine learning, the general way to train a model is to minimize the loss function $L(\boldsymbol{\beta})$ with respect to parameters $\boldsymbol{\beta}$. As in deep learning, models are so complex that we have no way to find the closed-form solution. Instead, we utilize gradient descent to find an optimal iteratively.

$$\beta_t = \beta_{t-1} - \eta \nabla L(\boldsymbol{\beta}_{t-1}) \tag{1}$$

Here, η is a constant called the learning rate. The choice of η is critical for training.

Firstly, we use constant learning rate to train the model without any weight decay. The learning rate is set to 0.1, 0.01 and 0.001 respectively. The networks is trained for 15 epochs. After 15 training epochs, the final losses and accuracy for the training set and the test set is shown in Table 1. The curve is plotted in figure 4.

learing rate	training loss	training acc	test loss	test acc
0.1	0.3711	87.13	0.4870	83.73
0.01	0.2225	92.12	0.3901	87.54
0.001	0.3554	87.63	0.5153	82.91

Table 1: The final losses and accuracy in terms of learning rate.

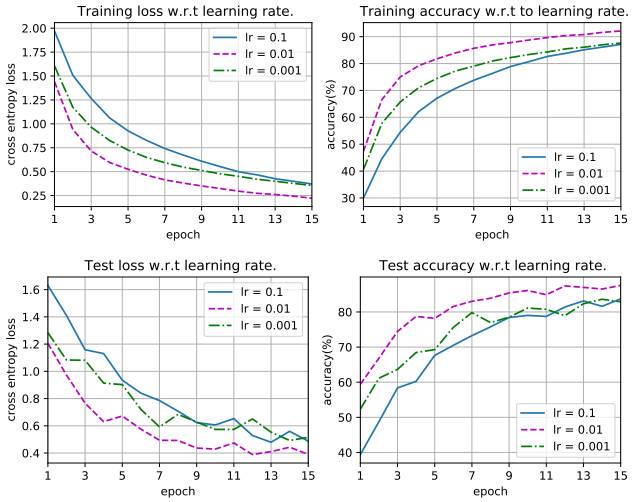

Figure 4: The training curves and test curves with respect to different learning rates.

It is indicated that the best learning rate is neither the largest one nor the smallest one. $\eta = 0.01$ becomes the best learning rate among these three in terms of training loss and training accuracy. $\eta = 0.01$ is also the best learning rate in terms of test loss and test accuracy. One possible reason is that a too small learning rate will cause the update very slow, while a too large learning rate will cause the update overshoots the optimal. Figure 5 gives an intuition for this reasoning.

2.2 Learning Rate Schedule

Next, we let the learning rate gradually decrease during the training. Mathematically, cosine annealing assigns the learning rate η_t with respect to dif-

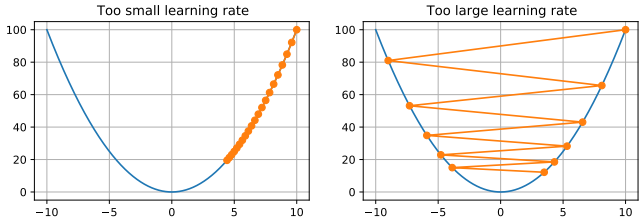

Figure 5: Toy example for different learning rate on gradient descent. The blue curve is on $y = x^2$. The orange curve is the trajectory.

ferent epochs t

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\frac{t}{T}\pi\right) \quad (2)$$

As the best learning rate $\eta = 0.01$ is discovered in the last subsection, let $\eta_{\max} = 0.01$. Typically, $\eta_{\min}$ is set 0. $T = 300$ allows learning rate gradually decrease during the whole training phase. Figure 6 shows exactly how the learning rate changes in this task by cosine annealing in the left and gives us an intuition towards how the learning rate scheduler works in the right. Compared with figure 5, we use the same large learning rate as the initial learning rate, but the difference is that the cosine schedule prevent the overshooting, especially near the optimal.

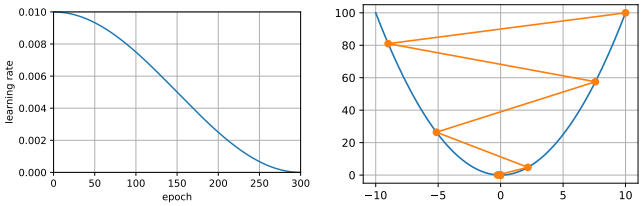

Figure 6: Left: The cosine annealing learning rate schedule. Right: Toy example with learning rate schedule.

In the experiment, I trained the model for 300 epochs with constant learning rate and with cosine annealing . Table 2 reports the final loss and accuracy after training. Figure 7 plots the learning curves. It is clear that training with cosine annealing achieves better result in all aspects. Scrutinizing the training loss curves, two curves almost overlap in the first 50 epochs, while the loss accelerate to decrease with cosine annealing. Why does smaller learning rate by cosine annealing lead to bigger training loss decrease? One possible reason is that when the parameters is already close enough to the optimal, large learning rate will only cause overshooting, so the training loss will not decrease as expected. In contrast, cosine annealing meets the fine tuning idea when the training loss has been close enough to the optimal.

scheduler type	training loss	training acc	test loss	test acc
constant	6.947e-04	99.98	6.947e-04	92.99
cosine annealing	1.020e-04	100.0	4.466e-01	93.66

Table 2: The final losses and accuracy in terms of learning rate schedulers.

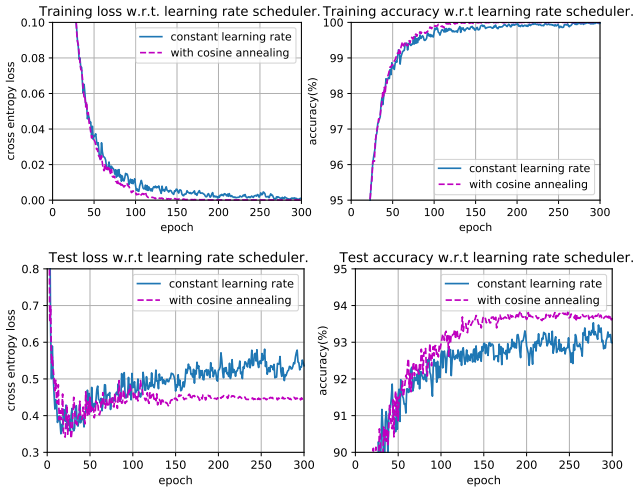

Figure 7: The training curves and test curves with or without scheduler.

2.3 Weight Decay

Up to now, the training accuracy has been near 100%, while the test accuracy still has space to improve. To avoid overfitting, weight decay is utilized to boost the model's generalization. Two different weight decay coefficients $\lambda_1 = 5 \times 10^{-4}$ and $\lambda_2 = 1 \times 10^{-2}$ were implemented.

Table 3 reports the final losses and accuracy after 300 epchs training. Figure 8 depicts the learning curve, which illustrates the effect of weight decays. The larger the weight decay coefficient is, the more slowly the training loss decrease.

weight decay	training loss	training acc	test loss	test acc
5e-4	1.338e-03	99.998	2.203e-01	94.6
1e-2	2.322e-02	99.994	1.735e-01	95.54

Table 3: The final losses and accuracy in terms of weight decay coefficients.

Figure 8: The training curves and test curves with different weight decay coefficients.

Surprisingly, although the model with smaller weight decay performs better on the test set at the first 250 epochs, the model with larger weigh decay finally overtakes it at the last 50 epochs. Another interesting point is that the training loss with 1×10^{-2} is still higher in the end, even though the test loss with 1×10^{-2} is lower.

2.4 Data Augmentation

Up to now, we have only use some basic data preprocessing techniques, like random horizontal flip, random cropping and normalization. In this subsection, a more advanced techniques called random erasing is applied to the training data. Figure 9 visualizes some samples after random erasing. Since the input pictures have been normalized in the previous experiment setups, the erasing value is just set to be 0.

The network was trained for 300 epochs. According to the table 10, the accuracy on the test set is improved, even though the accuracy on the training set is increased. It proves that data augmentation

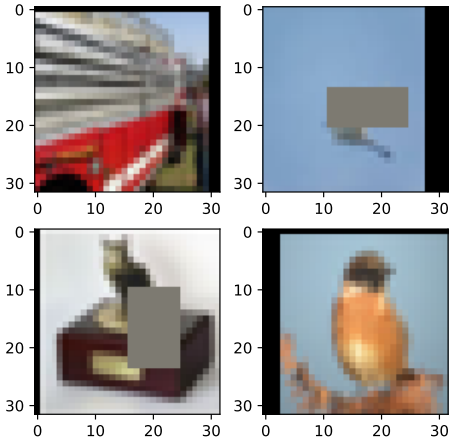

Figure 9: Some sample training images with data augmentation. The gray rectangle is due to random erasing. The probability of applying random erasing is 0.5, so no all images contains gray rectangle.

is an useful way to prevent overfitting and boost the model's generalization ability

data augmentation	training loss	training acc	test loss	test acc
no random erasing	2.322e-02	99.994	1.735e-01	95.54
random erasing	5.336e-02	98.994	1.364e-01	96.18

Table 4: The final losses and accuracy in terms of random erasing.

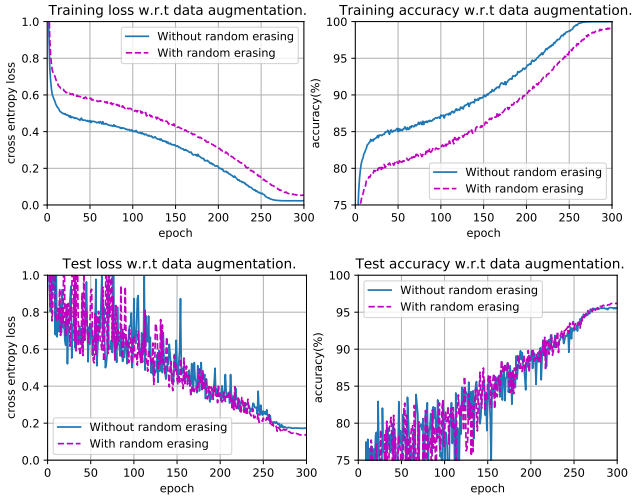

Figure 10: The training curves and test curves in terms of data augmentation.

References

[1] Google. Cuda convnet. https://code.google.com/archive/p/cuda-convnet/.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 7, 12 2015.

[3] Alex Krizhevsky. Cifar-10 and cifar-100 datasets. https://www.cs.toronto.edu/ kriz/cifar.htm.

[4] Sunqiande. https://blog.csdn.net/sunqiande88/article/details/80100891.

[5] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.