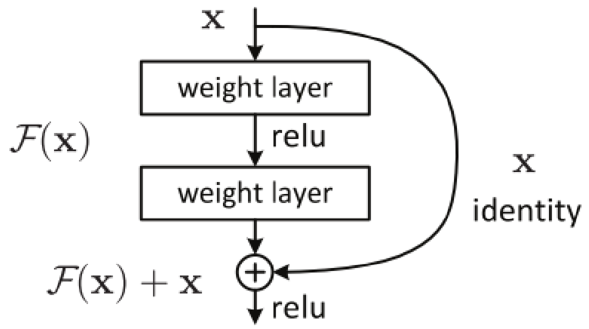
The Effects of Hyperparameters

Zheng Weixiang
School of Computer Science and Engineering
wzheng014@e.ntu.edu.sg

In this paper, we will investigate the effects of hyperparameters such as initial learning rate, learning rate scheduler, weight decay, and data augmentation on deep neural networks. This report includes the following parts: 1) The introduction of the model we are using: ResNet-18 [He et al.2015] and the dataset: CIFAR-10 [Krizhevsky, Hinton, and others2009]. 2) The effect of different initial learning rates. 3) The introduction of Cosine Annealing [Loshchilov and Hutter2016] and the difference made by using Cosine Annealing. 4) The effect of Weight decay and the effect of different values of weight decay. 5) The difference made by random erasing [Zhong et al.2020] data augmentation method.

ResNet-18 [He et al.2015]

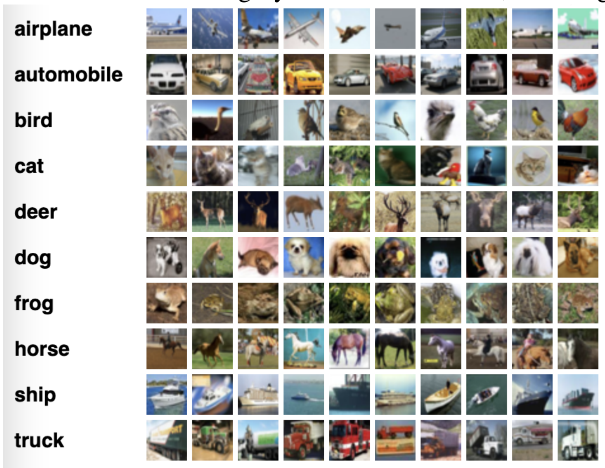
ResNet-18 is a convolutional neural network model introduced by He Kaiming in 2015. The basic components that build up the network are called the residual blocks. Every residual block utilizes skip connections to jump over some layers, which makes it possible to train much deeper than other models. Efficiently trained networks can have a depth of 100 layers and 1000 layers. Some common ResNet models use double-layer skips or triple-layer skips that contain nonlinearities(ReLU Layer) and batch normalization in between.


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

The main problem that ResNets have solved is Vanishing Gradients. When the network gets too deep, the gradients of loss function is easily shrink to zero after several applications of the chain rule. Then the weights never get updated, no learning is performed.With ResNets, however, the gradients can flow directly through the skip connections from later layers to initial layers.

CIFAR-10 [Krizhevsky, Hinton, and others2009]

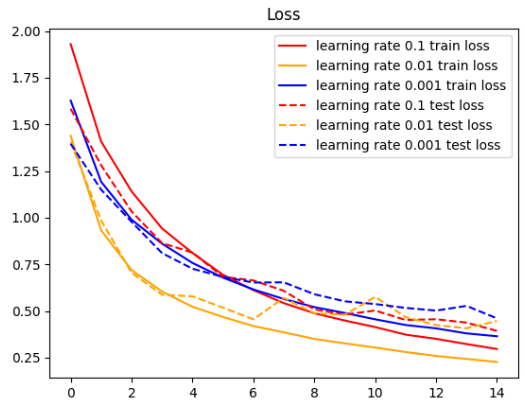
CIFAR-10 database(Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used image database for the purpose of training object recognition models. It contains 60,000 colored images labeled in 10 categories, each with a resolution of 32 pixels by 32 pixels. The 10 different categories are: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each category has a number of 6,000 images.

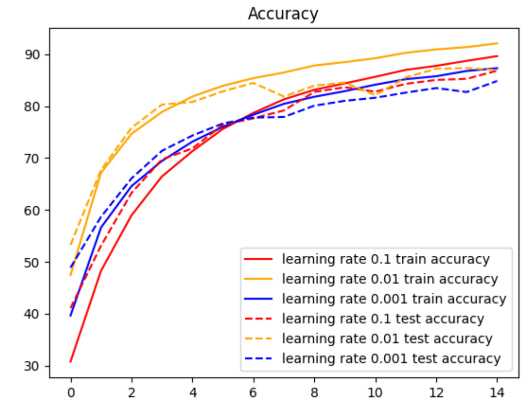


Almost all of the objects in the images are centered, which means without data augmentation, the trained algorithm will only recognize the object if it is placed in the center. Also, I notice in the horse class, for example, most horse images are place in such a way that the horse head is on the left and the horse tail is on the right, which I think will result in a biased algorithm, if without data augmentation. All the objects in images are clear and there's no occluded objects, for which I think if there exist some sort of blocking, the algorithm should still be able to recognize the object, and it could be done using data augmentation by cropping some parts out. All images contain no other objects, which I think is good, if there exist other objects, it might cause confusion for algorithm to learn the correct mapping.

Initial Learning Rate

In this section, we briefly discuss the effect of different initial learning rates to our final result. The settings are as follows: The batch size is set to 128 for both Training and Testing set. No Weight decay or Learning Rate Scheduler is used. For data augmentation, only random cropping and random horizontal flip is used. We will run tests on three different learning rates: 0.1, 0.01, and 0.001. For every learning rate setting, 15 epoches will be run to train the neural network. The attached figures are the Loss plot and Accuracy plot. We can see from the plot that model with learning rate of 0.01 has the lowest training loss. I think this is because a model with excessively large learning rate will converge quickly to a sub-optimal point.



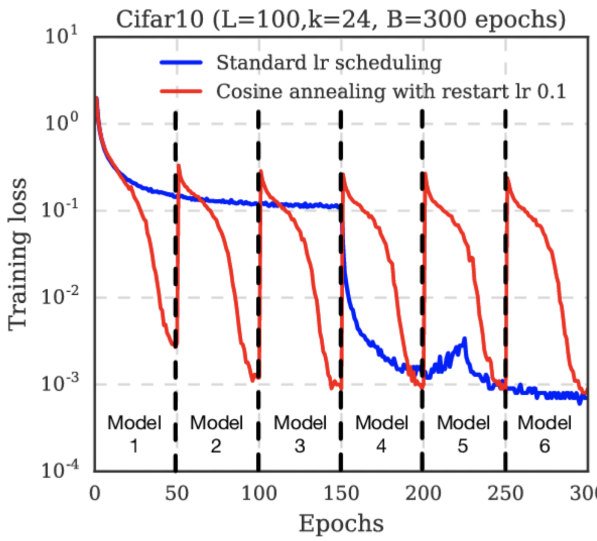


On the other hand, a model with a overly small learning rate will have a hard time proceeding towards the optimal point therefore get stuck at high loss position. Also, we notice the test loss of learning rate 0.1 model overtake the first position of learning rate 0.01 model at the end. I suspect there might be several reasons: 1. Train-Test dataset inconsistency, maybe training dataset and testing dataset didn't get shuffled randomly enough, because our model is trained on training set and tested on testing set, the inconsistency could make model perform good on training set and not so well on testing set. 2. there wasn't much sub-optimal points for large learning rate model to be trapped, this could due to the fact that training set is generic enough. As for the accuracy, we can see from the plot that the model with learning rate of 0.01 is leading the competition all the way until the end. However, at around epoch 7, the testing accuracy starts to fluctuate. I guess this could because of the fact that model before epoch 6 was learning all the generic characteristics about the training data which is also hold for the testing data, but at the exact time of epoch 6, it learnt some characteristics that is so specific to some training data that does not generalize well on the testing set. Normally speaking, it could be a sign of overfitting, but since after epoch 6, the testing accuracy is still going up, I suspect it was by accident that model learnt some training-specific characteristics. To conclude, overall the model with learning rate of 0.01 performs the best because it is not too large and not too small. The model with learning rate of 0.1 is also worth mentioning since it has smaller testing loss than the model with learning rate of 0.01.

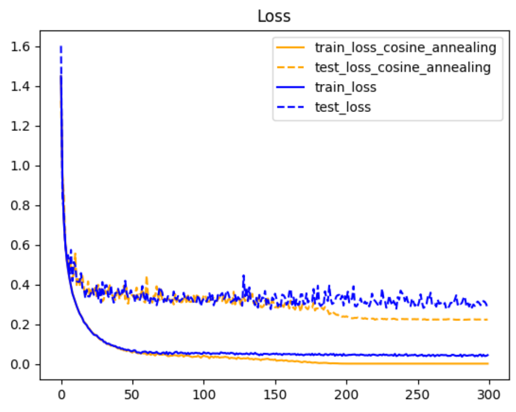
Learning Rate Scheduler

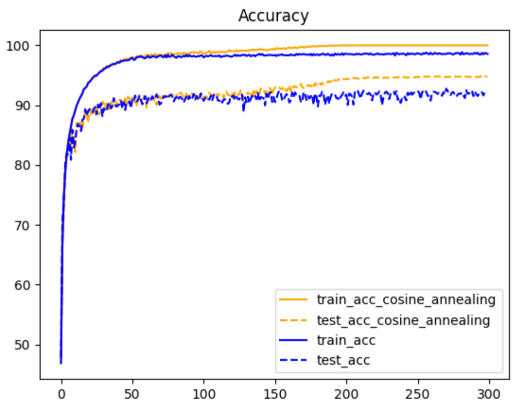
In this section, let's have a look at the Cosine Annealing Learning Rate Scheduler. Cosine Annealing Scheduler is a type of learning rate scheduler that starts with a large learning rate and quickly decrease to a minimum value before quickly increase again. The resetting of the learning rate simulates the process of restarting the learning and re-use the trained weights as the starting point. It is referred as "warm restart". The formula [Cos] and the figure [Cos] are as follows:

$$\eta_t = \eta^i_{min} + \frac{1}{2}\left(\eta^i_{max} - \eta^i_{min}\right)\left(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right)\right)$$



Notice how the cosine function is used in the formula and therefore the naming. It has been shown to perform better than its alternative like Simple Linear Annealing in practice. [Cos] Then we will briefly talk about the effect of different learning rate scheduler to the final result. The settings are as follows: The batch size is set to 128 for both Training and Testing set.Momentum is set to 0.9. Weight decay of 5e-4 is used. For data augmentation, only random cropping and random horizontal flip is used. We will run tests on two different models: with Cosine Annealing and without Cosine Annealing. For every model, 300 epoches will be run to train the neural network. From the loss plot, we can see the difference at the beginning is not very clear, but as we run more epoches, the model without Cosine Annealing starts to show a sign of overfitting whereas the loss of model with Cosine Annealing keeps down. This is because in the end, the learning rate becomes relatively large for the model without Cosine Annealing and therefore overshoot occurs. The model fails to converge because every step is just too large to make. As we run more epoches, in the model with Cosine Annealing, learning rate will be very close to 0, and the specific characteristics about this training set/training set noise won't contribute so much as the previous data points. However, in the model without Cosine Annealing because we keep the learning rate fixed, the model begins to learn training set specific characteristics at the same rate as before. Therefore we see a very clear trend of overfitting at a stable rate, the loss goes up.

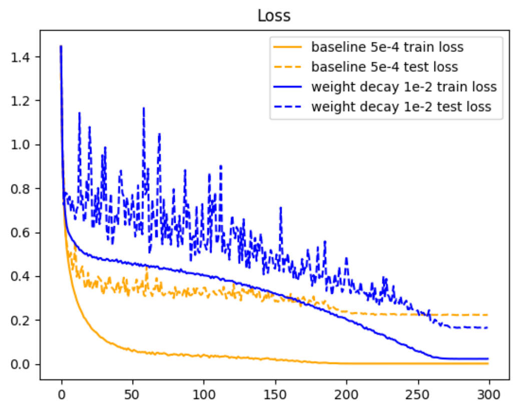




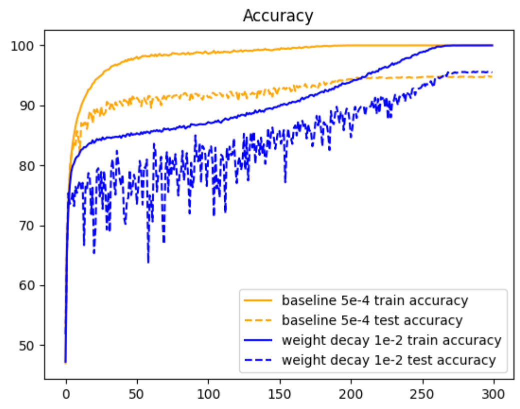
The accuracy agrees with the loss. At the beginning the difference is negligible. From the same epoch where model without Cosine Annealing starts to overfit, the accuracy starts to drop as well. The conclusion is therefore dynamically adjusting learning rate is a better option than fixed learning rate.

Weight Decay

In this section, we will briefly talk about the effect of different weight decay methods to the final result. The settings are as follows: The batch size is set to 128 for both Training and Testing set.Cosine Annealing is used. For data augmentation, only random cropping and random horizontal flip is used. We will run tests on two different models: baseline model with weight decay of 5e-4 and with weight decay of 1e-2. For every model, 300 epoches will be run to train the neural network. We can see from the loss plot that baseline model(the best model so far, with cosine annealing, weight decay 5e-4, momentum 0.9, learning rate 0.01) also perform the better. In terms of training loss, I would say the difference is very small, this is due to the fact that we run 300 epoches and that is enough to lower the training loss no matter how strong the regulating strength is. The difference between their testing loss, however, is significant. I suspect this is due to the fact that model with 1e-2 has a much stronger(magnitude of 2) weight decay coefficient than the baseline model, so in terms of the final model, the model with weight decay coefficient of 1e-2 will be strictly more compact than the other model, i.e. the baseline model will be more complex, the weights in baseline model will be higher in term of numerical values, therefore more overfitting than the other one.

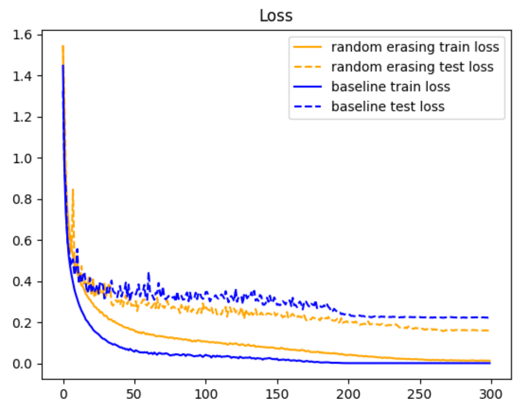


The effect of light weights is also shown in accuracy plot. We can see that both model reaches 100% accuracy for training, but for testing, the model with stronger weight decay coefficient therefore lighter weights outperforms the baseline model. However, another thing to notice is that we clearly observe more fluctuation for higher weight decay coefficient model, both in loss plot and accuracy plot. I think this is because stronger weight decay coefficient cause the model weights to change more therefore affect the overall loss/accuracy more than the other model, in both good and bad ways, therefore more obvious spikes. We also notice the convergence of loss and accuracy of stronger weight decay model takes more epoches than the baseline model, this is because more fluctuation means more time to converge.

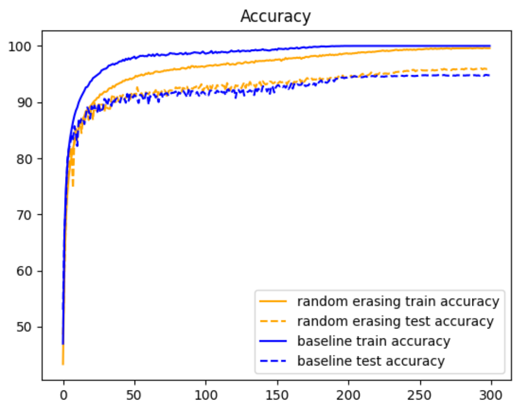


Data Augmentation

In this section, we will briefly talk about the effect of different data augmentation methods to the final result. The settings are as follows: The batch size is set to 128 for both Training and Testing set.Cosine Annealing is used. Weight Decay of 5e-4 is used. We will run tests on random erasing augmentation [Zhong et al.2020].300 epoches will be run to train the neural network. Then we will compare the result with result of the best model(RandomHorizontalFlip and RandomCrop) so far. Just to make things more clear, the random erasing model also has RandomHorizontalFlip and RandomCrop, it has an additional random erasing compared to the baseline model, other things being equal. From the loss plot, we can see that the train losses of different models eventually meet at the end, however, the numerical values are quite different, with baseline model having a loss of 0.001344 and random erasing model having a loss of 0.013064. The difference is ten fold. One possible reason is that the random erasing in the transformation created some confusion for the model to learn, some random erasing might happen to erase the most important part, i.e. the plane to be recognized in a plane image. The testing loss, however, disagree with the training loss, with baseline model achieved a loss of 0.2242 and random erasing model achieved a loss of 0.1608. This is due to the fact that we trained the model using occluded data, forcing the model to learn some generic characteristics of the object, the effect of prevent overfitting is better than the baseline model. In the loss case, baseline has a training loss lower than random erasing model, but a testing loss higher than random erasing model. The baseline model is overfitting. Another interesting point is that the baseline model learns the feature more quickly at the beginning, this is because there is no occlusion, the model can quickly locate the determining feature for recognizing the object. With occlusion, the model has to weigh over some other features since the determining feature might not be avilable all the time therefore slower for loss curve to decrease, but thank to the generic feature it learnt during the training, the testing loss curve will decrease faster than the baseline model, because the test data is not seen before, the generic feature it learnt makes it at advantage.



As for the accuracy plot, the baseline model also shows a sign of overfitting, i.e. higher training accuracy, lower testing accuracy. The baseline model is better specifically at the training set means it is not generic enough and learnt some noise of the training set. The numeric accuracy of training accuracy at the end is 99.606 for random erasing model and 100 for baseline model. The numeric accuracy of testing accuracy at the end is 95.86 for random erasing model and 94.78 for baseline model. The conclusion is that random erasing method can reduce overfitting, therefore leads to a lower testing loss and a higher testing accuracy. The disadvantage is that it takes longer for the model to converge.



Conclusion

In this paper, we had a look at the architecture of ResNet-18, quickly go over the CIFAR-10 dataset, and had several experiments with different hyperparameters. The hyperparameters we experimented includes initial learning rate, learning rate scheduler, weight decay coefficient, and data augmentation. In initial learning rate experiment, we tested with different rates of 0.1, 0.01, and 0.001. We found 0.01 is the best learning rate. We conclude learning rate need to be adequate, being too large and being too are both bad for the performance. In learning rate scheduler experiment, we found Cosine Annealing scheduler is good at boosting performance, for both training and testing process. In weight decay experiment, we found a larger weight decay coefficient causes stronger regularization, leads to a more compact model(smaller weights) therefore prevent overfitting. However, weight decay also causes the model to converge more slowly. In data augmentation experiment, we found random erasing is a good method to force model learn the generic feature and therefore prevent overfitting. The down side is the same as weight decay, at a cost of slower convergence.

References

Cosine annealed warm restart learning schedulers. https://www.kaggle.com/residentmario/cosine-annealed-warm-restart-learning-schedulers. Accessed: 2022-03-16.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *CoRR* abs/1512.03385.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Loshchilov, I., and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2020. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 13001–13008.