AI6103 Homework Assignment Report

Yu Yue
School of Computer Science and Engineering
50 Nanyang Ave, #32 Block N4
Singapore 639798
yyu025@e.ntu.edu.sg

Abstract

The effects of hyperparameters, such as initial learning rate, learning rate schedule, weight decay, and data augmentation on deep neural networks are investigated in our assignment work. The ResNet-18 (He et al. 2015) network and the CIFAR-10 dataset are mainly used.

Data Characteristics

For the dataset CIFAR-10, we show some images to develop a better sense of the data. We split the CIFAR-10 (Wikipedia contributors 2022) training set into a new training set and a validation set which containing 40,000 and 10,000 data points respectively. And then we show the first batch of the new training set which contains 128 32×32 images in the figure 1. As we can see from the images:

- Some objects in the image are occluded. (e.g., the car image in row 1 column 14, there are two people sitting on the engine hood.)
- There are some other objects in the images. (e.g., the horse image in row 7 column 12, there is a man on a horse)

To partition the original training set with $50,000$ images into a new training set with $40,000$ images, and a validation set with $10,000$ images, the lines of code used are the following:

```
import torch
# set the random seed 0 for the partitioning
torch.manual_seed(0)
train_set_size = 40000
val_set_size = 10000
train_set, val_set = torch.utils.data.random_split \
        (dataset, [train_set_size, val_set_size], \
        generator=torch.Generator().manual_seed(0))
```

We also show the proportion of each class in the new training set in the table 1. As we can see from the table, for each class in the new training set and the validation set, the proportion is all around 10%, which means the datasets follow class balance.

number and proportion of each class		
class name	training set	validation set
airplane	4014 (10.04%)	986 (9.86%)
automobile	4002 (10.01%)	998 (9.98%)
bird	3995 (9.99%)	1005 (10.05%)
cat	3977 (9.94%)	1023 (10.23%)
deer	3982 (9.96%)	1018 (10.18%)
dog	4019 (10.05%)	981 (9.81%)
frog	3975 (9.93%)	1025 (10.25%)
horse	4010 (10.03%)	990 (9.90%)
ship	4010 (10.03%)	990 (9.90%)
truck	4016 (10.04%)	984 (9.84%)
average	4000	1000
variance	252	252

Table 1: number and proportion of each class in the new training and validation set

Learning Rate

In the experiments, we train ResNet-18 model on the training set with $40,000$ images for 15 epochs with different learning rate (0.1, 0.01 and 0.001). The batch size is set to 128 and we use neither weight decay nor learning rate schedule. The final losses and accuracy values for both the training set and the validation set are plotted in the training curve graphs shown in the figure 2.

From the training curves, we found for each experiment, the training loss and the validating loss tend to decrease, while the training accuracy and the validating accuracy tend to increase. However, after training 15 epochs, we found that when the learning rate is 0.01, the training loss tends to 0.2463 while the training loss only achieves 0.4601 and 0.4035 when the learning rate is 0.1 and 0.001. The validating loss is also lower when the learning rate is 0.01, which reaches 0.4309. But it only reaches around 0.5620 and 0.5610 while the learning rate is 0.1 and 0.001. Correspondingly, the accuracy for the training set and the validation set achieve 91.39% and 86.06% when the learning rate is 0.01, while it only achieves 83.87% for training set and 81.31% for validation set when the learning rate is 0.1, and achieves 85.81% for training set and 80.78% for validation set when the learning rate is 0.001.

In summary, when the learning rate is 0.01, the loss for

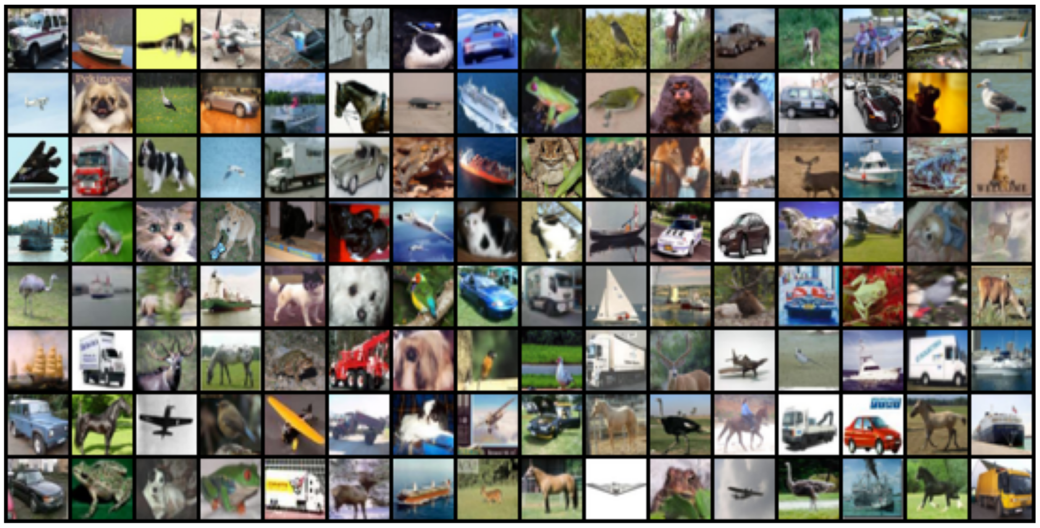

Figure 1: First batch of the training set containing 128 32×32 images

training and validation sets are both lower than those with learning rate 0.1 or 0.001 and the accuracy for training and validation sets are both higher than those with learning rate 0.1 or 0.001 after training 15 epochs. Which means the training converges relatively faster when the learning rate is 0.01. The reasons are:

- The learning rate determines how much the parameters change at each step.
- If the learning rate is too small (i.e., 0.001), the convergence becomes slower because we move too small at each step. Then, training for the same number of epochs may not achieve convergence.
- If the learning rate is too large (i.e., 0.1), It may lead to oscillation and overshooting. We move too far at each step.

Therefore, when the learning rate is 0.001, for each iteration, the parameters of the model changes too small, then it achieves convergence slower; when the learning rate is 0.1, for each iteration, the parameters of the model changes too large, then it may lead to oscillation, which also slows down the convergence speed. Therefore, before the convergence, the training process with learning rate 0.01 achieves better performance because it is closer to the convergence after training 15 epochs.

Learning Rate Schedule

Instead of the constant learning rate, we want dynamic learning rate because at the beginning, we want larger learning rate for faster convergence, and then smaller learning rate to avoid oscillation and overshooting. There we introduce cosine annealing. For the learning rate η_t with the current number of epochs T_{cur} and the total number of epochs T_{max}:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{cur}}{T_{max}}\pi))$$

(Loshchilov and Hutter 2016)

In our work, we set η_{min} as 0 and η_{max} as the original learning rate 0.01. We train the model for 300 epochs:

$$\eta_t = \frac{1}{2} \times 0.01 \times (1 + cos(\frac{T_{cur}}{300}\pi))$$

At the beginning, the learning rate $\eta_t = \eta_{max} = 0.01$, and at the end $\eta_t = \eta_{min} = 0$, The curve of the learning rate corresponding to the epoch number is shown in figure 3. Then, we conduct two experiments, one with the constant learning rate 0.01 and the other with cosine annealing which decreases the initial learning rate from 0.01 to 0 over the entirety of the training session, both with 300 epochs training. The learning curves of the loss and the accuracy are shown in the figure 4.

As we can see from the results, with or without cosine annealing, the training loss decreases to around 0 and the training accuracy comes to around 100% after 300 epochs for both of two experiments. However, if we observe the accurate values, we find the training loss without cosine annealing schedule oscillates between 0.001 and 0.002 in the final several epochs, while the training loss perfectly achieves 0.0001 with cosine annealing. Similarly, for the training accuracy, the one without cosine annealing also oscillates between 99.90% to 99.98% while the one with cosine annealing perfectly comes to 100.0%. For the validation set, as we

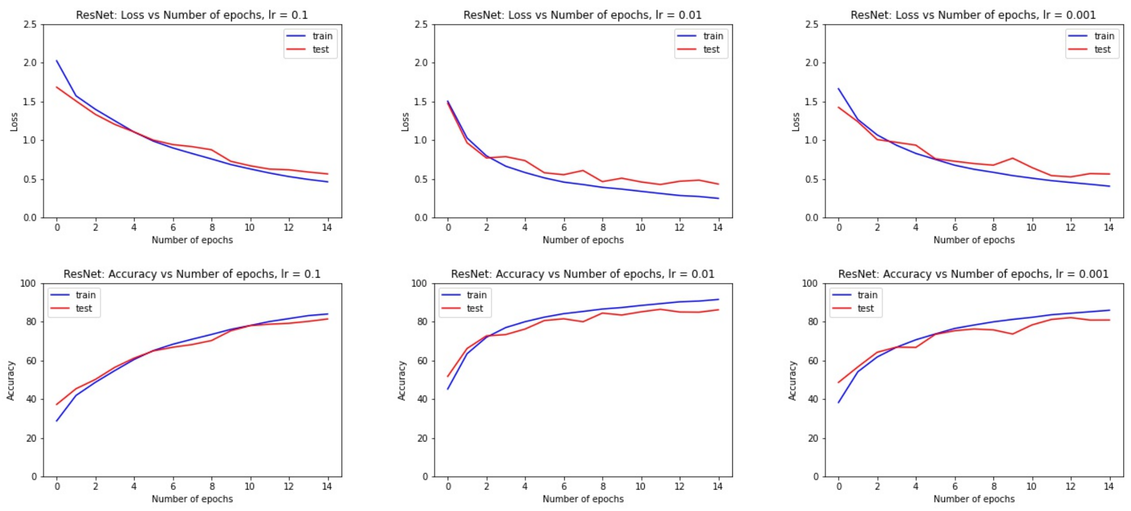

Figure 2: The loss (top row) and the accuracy (bottom row) curves for different learning rate 0.1 (left), 0.01 (middle) and 0.001 (right). The x-axis is the number of the epochs. We train 15 epochs for each.

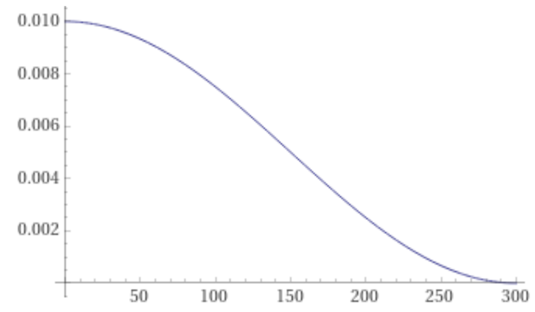

Figure 3: learning rate (y-axis) with different epoch number (x-axis) by using cosine annealing schedule

.

	train		validation	
	no CA	CA	no CA	CA
loss	0.0019	0.0001	0.5766	0.5065
accuracy	99.94%	100.0%	91.91%	92.45%

Table 2: the loss and the accuracy values for training and validation set with or without cosine annealing (CA) at epoch 300.

can see from the graphs, the oscillation for both loss and accuracy values when no cosine annealing are much greater than the one with cosine annealing. Consequently, with cosine annealing, we can achieve lower and stable training and validating loss, and higher and stable training and validating accuracy shown in the table 2.

The reasons why the performance with cosine annealing is stable and better is with cosine annealing, the learning rate decreases during the whole training process. At the beginning, the learning rate is relatively large, so that gradient descent can be performed faster, then lower loss can be achieved in fewer epochs. Then, the learning rate is gradually reduced during the learning process, so as to reduce the oscillation for faster gradient descent. At the end of the training, the learning rate is relatively small to avoid overshooting, and reaches ”local minimum”. However, without cosine annealing, the learning rate is a constant, then at the end of training, the parameters changes too large at every step, which leads to oscillation and overshooting.

Weight Decay

We then introduce weight decay to the training process to the best experimental setting discovered so far (learning rate is 0.01 with cosine annealing schedule). With two different weight decay coefficients $\lambda = 5 \times 10^{-4}$ and 1×10^{-2}, the training curves are shown in the figure 5.

As we an see from the training curves, with the weight decay coefficient $\lambda = 1 \times 10^{-2}$, the downward trend of the training loss curve has slowed, which near the convergence until around 250 epochs, and the training accuracy is also rising much slower, reaches around 100.0% till 250 epochs. For the validation loss curve and the accuracy curve, the oscillations are more violent before 250 epochs, but it reaches relatively higher at the end. With the weight decay coefficient $\lambda = 5 \times 10^{-4}$, the magnitudes of the oscillations of the training loss and the training accuracy decrease, and the convergence is relatively faster (around 200 epochs). For the validation loss curve and the accuracy curve, the oscillations are also less violent, but it reaches relatively lower as we can see from the table 3.

Because weight decay is a method for regularization (Zhang et al. 2018). If there is no weight decay, the model tend to memorize all the training set, which leads to overfitting, and then the performance on the validation set is worse. That’s why with weight decay, the performance on the validation set becomes better. However, if the weight decay coefficient is too large, which means the parameters are too

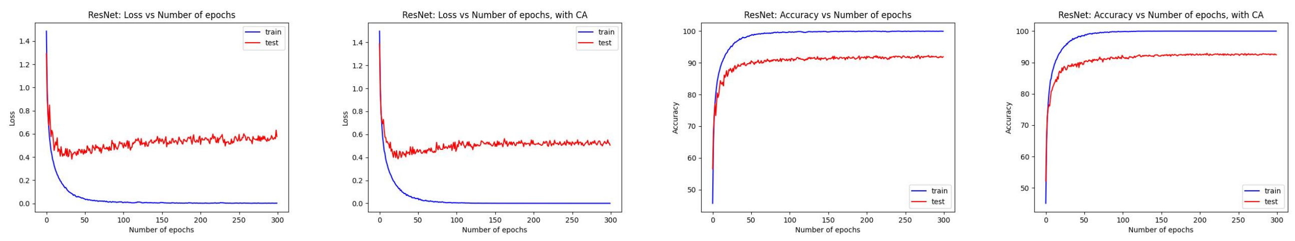

Figure 4: The loss (left two) and the accuracy (right two) curves for training without cosine annealing (left) and with cosine annealing (right). The x-axis is the number of the epochs. We train 300 epochs for each. The learning rate = 0.01.

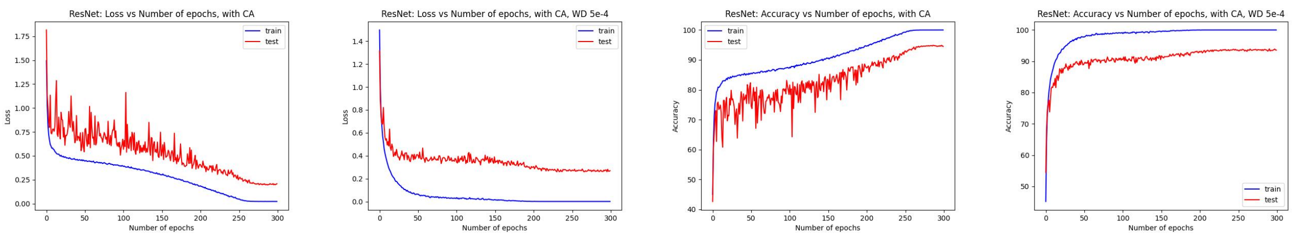

Figure 5: The loss (left two) and the accuracy (right two) curves for training with weight decay coefficient $\lambda = 1 \times 10^{-2}$ (left) and $\lambda = 5 \times 10^{-4}$ (right). The x-axis is the number of the epochs. We train 300 epochs for each. The learning rate = 0.01 with cosine annealing.

	train		validation	
wd	$1e-2$	$5e-4$	$1e-2$	$5e-4$
loss	0.0230	0.0012	0.2078	0.2696
accuracy	100.0%	100.0%	94.51%	93.57%

Table 3: the loss and the accuracy values for training and validation set with weight decay coefficient $\lambda = 1 \times 10^{-2}$ or 5×10^{-4} at epoch 300.

small and the regularization effect is relatively stronger, then it leads to underfitting, and the model may fail to converge. That's why when $\Lambda = 1 \times 10^{-2}$, there are more oscillations, and the convergence process becomes slower. When the weight decay coefficient $\Lambda = 5 \times 10^{-4}$, the model has a strong fitting ability and can converge quickly at the beginning of training. At the same time, the model has a high generalization ability and can achieve a better performance on the validation set.

Data Augmentation

In this chapter, we experiment with the cutout augmentation technique using *torchvision.transforms.RandomErasing* (DeVries and Taylor 2017) with the best experimental setup discovered so far (with cosine annealing, learning rate = 0.01 and weight decay $\lambda = 5 \times 10^{-4}$). We set the erasing value in the parameter **value** as the respective means of the three color channels which calculated across the entire training set to minimize the effects of the augmentation on the image distribution. Firstly, we set the probability that the random erasing operation will be performed **p** as 1.0, the range of proportion of erased area against input image **scale** as $(0.02, 0.33)$ and the range of aspect ratio of erased area **ratio** as $(0.3, 3.3)$. The training curves are shown in figure 6. The final losses and accuracy values for both the training set and the validation set are shown in the table 4.

	train	validation	test
loss	0.0125	0.1944	0.1842
accuracy	99.64%	94.62%	95.11%

Table 4: the loss and the accuracy values for training, validation and test set with coefficient $p = 1.0$, $scale = (0.02, 0.33)$ and $ratio = (0.3, 3.3)$ at epoch 300.

Compared with the training curves with or without random erasing, we find that with random erasing, the validation accuracy is higher than those without random erasing, and the validation loss is also slightly lower than those without random erasing. However, the training accuracy doesn't achieves 100% at epoch 300, and the training loss is also slightly higher. The training curves are similar to the previous one. But the the model converges at around 250 epoch, which is slower than those without random erasing.

That's because if we have data augmentation, there are more noises in the training data. We force the model not only focus on the main part of the objects in the images by using random erasing, then the model really learns the characteristics of objects but not just one part. That's why the performance on the validation set improves, but the training process convergence becomes slower. Because data augmentation like random erasing is a regularization method, the training accuracy at the epoch 300 doesn't achieves 100.0% as before, but it actually increases the generalization ability

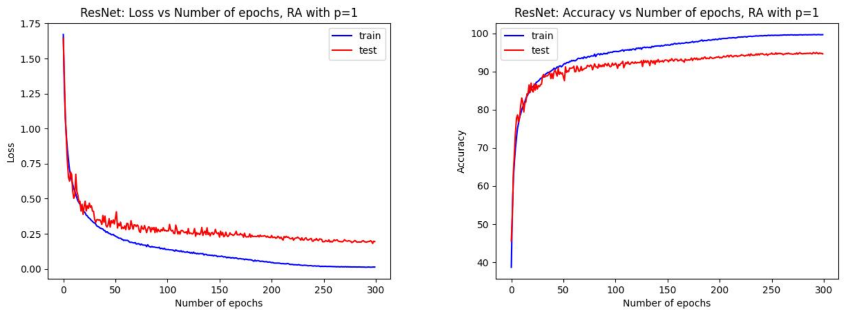

Figure 6: The loss (left) and the accuracy (right) curves for training with random erasing with coefficients $p = 1.0$, $scale = (0.02, 0.33)$ and $ratio = (0.3, 0.33)$. The x-axis is the number of the epochs. We train 300 epochs for each. The learning rate = 0.01 and the weight decay $\lambda = 5 \times 10^{-4}$ with cosine annealing.

of the model. As we can see from the table 4, the accuracy on the hold-out test set is 95.11%. We believe the trained model to perform at for all similar images in the future is very accurate because of its generalization ability brought by random erasing.

Data Augmentation with Fine-tuning

In this chapter, we tune the hyperparameters of the cutout augmentation to see the different effects. The parameters we focus on are the probability that the random erasing operation performed **p**, the range of proportion of erased area against input image **scale** and the range of aspect ratio of erased area **ratio**.

Probability *p*. We set the different probabilities that the random erasing operation performed $\mathbf{p} = 0.8, 0.5$. The training curves are shown in the figure **??** and the final results are shown in table 5. As we can see from the training curves and the specific loss and accuracy values, with $p = 1.0, 0.8, 0.5$, the training results are close to each other. The accuracy on the test set achieves around 95% for three experiments.

Proportion of erased area *scale*. We also tune the hyperparameter **scale** to show the effects of the range of proportion of erased area against input image. We set **scale** $= (0.02, 0.33)$ previously, ad we increase this proportion to **scale** $= (0.17, 0.50)$ and **scale** $= (0.33, 0.67)$ to see the effects of increasing the erased area. For the other parameters, we set $\mathbf{p} = 1.0$, **ratio** $= (0.3, 3.3)$, learning rate = 0.01, $\lambda = 5 \times 10^{-4}$ with cosine annealing. The training curves are shown in figure 8 ad the loss and accuracy values are shown in the table 6.

We found when the erased area becomes larger, the convergence process becomes slower, and the accuracy for the training set also becomes lower. When the erasing areas are too large, the erasers make too much noise in the data, so that the side effects outweigh the regularization effect, which may lead to underfitting.

Aspect ratio of erased area *ratio*. We tune the hyperparameter **ratio** to show the effects of the aspect ratio of erased area. We set **ratio** $= (0.3, 3.3)$ previously, ad we set to **ratio** $= (0.9, 1.1)$ and **ratio** $= (0.2, 5.0)$ to explore which effect is better: the erasing closer to the square or more narrow erasing area? For the other parameters, we set $\mathbf{p} = 1.0$, **scale** $= (0.02, 0.33)$, learning rate = 0.01, $\lambda = 5 \times 10^{-4}$ with cosine annealing. The training curves are shown in figure 9.

We found when the erased area becomes narrow, the performance is very close to each other, we cannot tell which one is much better. But they both achieves comparable performance. The loss and accuracy values are shown in the table 7.

References

DeVries, T.; and Taylor, G. W. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

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

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

Wikipedia contributors. 2022. CIFAR-10 — Wikipedia, The Free Encyclopedia. [Online; accessed 5-October-2022].

Zhang, G.; Wang, C.; Xu, B.; and Grosse, R. 2018. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*.

	train		validation		test	
p	0.8	0.5	0.8	0.5	0.8	0.5
loss	0.0123	0.0126	0.2081	0.1955	0.1990	0.1923
accuracy	99.63%	99.61%	94.64%	94.74%	94.76%	94.96%

Table 5: the loss and the accuracy values for training, validation and test set with coefficient $p = 0.8$ and $p = 0.5$ at epoch 300.

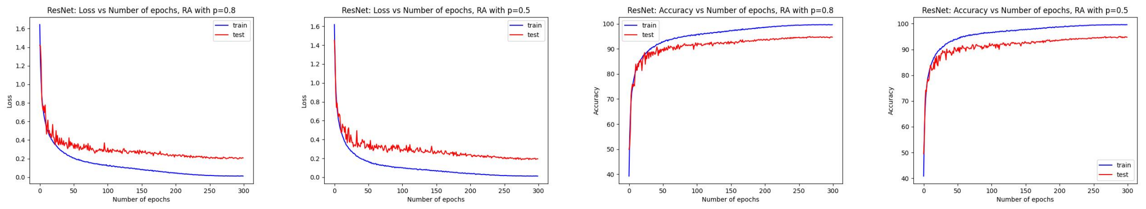

Figure 7: The loss (left two) and the accuracy (right two) curves for training with random erasing with coefficient $p = 0.8$ (left) and $p = 0.5$ (right).

	train		validation		test	
scale	$0.17 - 0.5$	$0.33 - 0.67$	$0.17 - 0.5$	$0.33 - 0.67$	$0.17 - 0.5$	$0.33 - 0.67$
loss	0.0327	0.0829	0.1986	0.2647	0.1980	0.2715
accuracy	98.96%	97.27%	94.33%	92.87%	94.98%	92.87%

Table 6: the loss and the accuracy values for training, validation and test set with coefficient $scale = (0.17, 0.33)$ and $scale = (0.33, 0.67)$ at epoch 300.

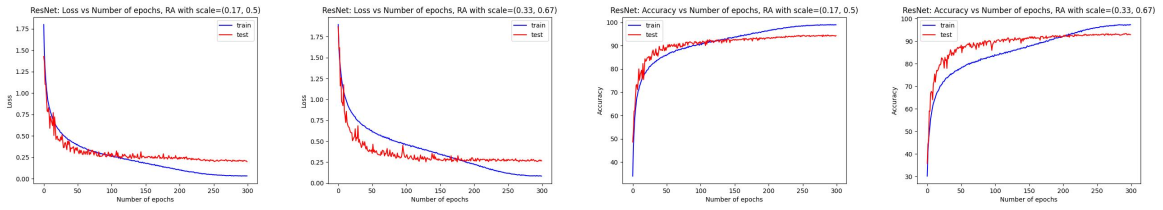

Figure 8: The loss (left two) and the accuracy (right two) curves for training with random erasing with coefficient $scale = (0.17, 0.5)$ (left) and $scale = (0.33, 0.67)$ (right).

	train		validation		test	
ratio	$0.9 - 1.1$	$0.2 - 5$	$0.9 - 1.1$	$0.2 - 5$	$0.9 - 1.1$	$0.2 - 5$
loss	0.0101	0.0114	0.2092	0.1946	0.1918	0.1843
accuracy	99.73%	99.67%	94.53%	94.75%	94.90%	95.02%

Table 7: the loss and the accuracy values for training, validation and test set with coefficient $ratio = (0.9, 1.1)$ and $ratio = (0.2, 5.0)$ at epoch 300.

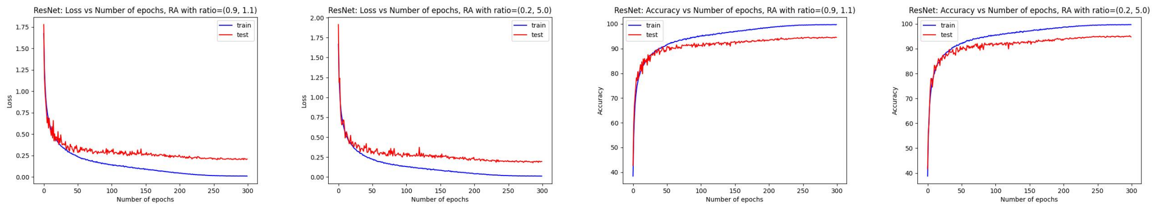

Figure 9: The loss (left two) and the accuracy (right two) curves for training with random erasing with coefficient $ratio = (0.9, 1.1)$ (left) and $ratio = (0.2, 5.0)$ (right).