Report for cifar-10 classifying task

Zhengye Bao

November 16, 2025

Abstract

In this project, I present an intermediate classifier that is trained from the cifar-10 dataset with an accuracy rate of 81.10% in its test set within 20 epochs. Moreover, I conducted several explorations into the model structure, data augment, and parameter tuning. Detailed information is as follows.

1 Methods

1.1 baseline

I select resnet, ViT, and the classical Alexnet as my baseline.

1.2 Model structure

For resnet, I designed a structure as 1 shows. For ViT, I designed 4 transformer layers whose FFN is 64 dimensions and head number is 4. Each patch tokenized is in 4x4 size. For Alexnet, I maintain the original structure without any changes(in following parts of this report, I would name Alexnet as cnn for simplicity).

1.3 hyperparameter tuning (for resnet)

default setting: optimizer=AdamW, initialization=He uniform

1.3.1 conv kernel size

There could be 5 different kernels in resnet, so I tried different strategies in 8 (each number s stands for sxs kernel size of the convolution layer respectively). According to the resnet paper, 73333 would be the best, as a broader receptive field might capture general information initially, while the final of feature extraction may require details. However, this trial failed, and surprisingly 55555 performed the best. Maybe that's because the recognition of images is so low that 7x7 kernel becomes relatively excessively enormous.

1.3.2 learning rate

I tried lr=0.0075,0.005,0.0025 and 0.002 and found little difference in f1score and accuracy between 0.0025 and 0.002. Finally, I selected lr=0.002, as it's slightly more stable. See 13.

1.4 optimizer

I considered AdamW, Adam, SGD and SGDM as optional optimizers. Though AdamW outperformed the others within 10 epochs, SGDM seemed to have not yet converged so I tried a larger learning rate=0.003, but it doesn't work. See 19.

1.5 initialization

I considered He uniform, Xavier uniform and Xavier normal. It turned out almost the same, but He uniform performed slightly better. See 23.

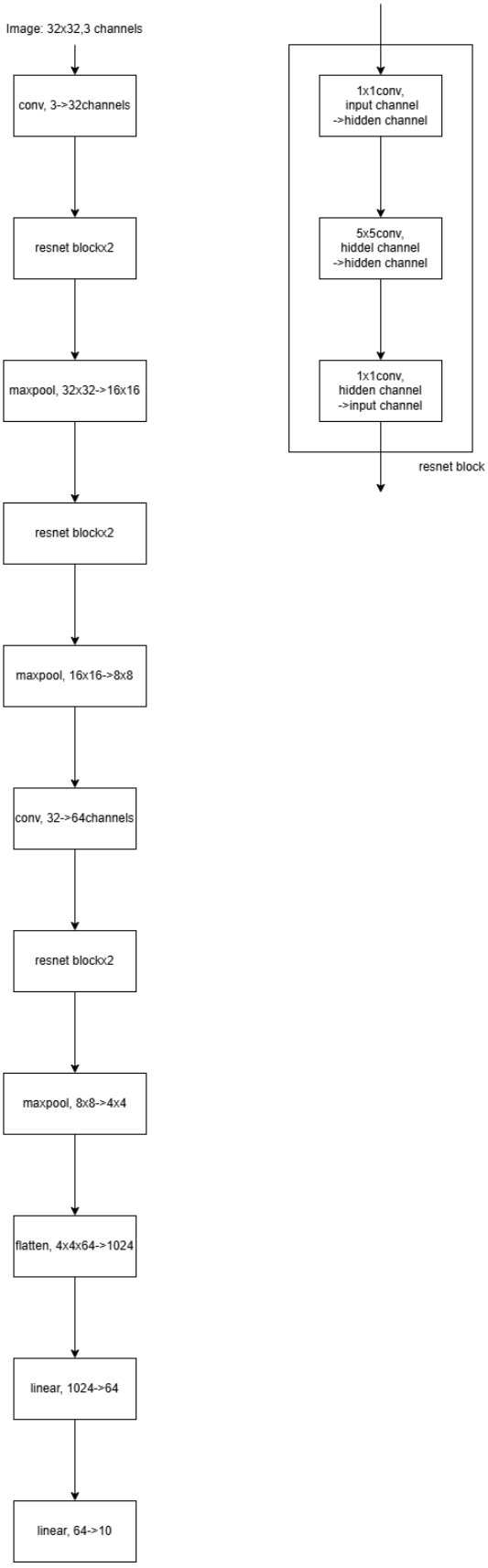

Figure 1: resnet structure

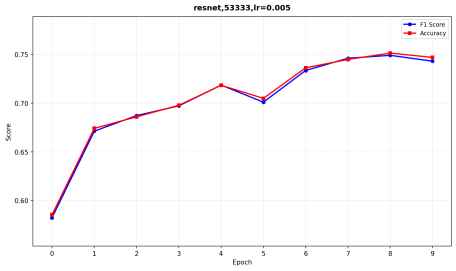

Figure 2: 53333,maxacc=0.7513,maxf1=0.7490

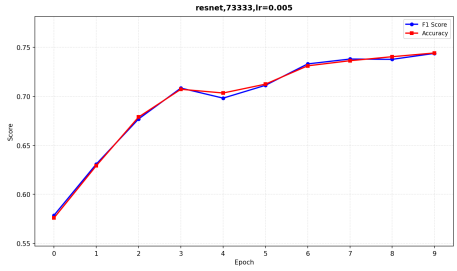

Figure 3: 73333,maxacc=0.7443,maxf1=0.7439

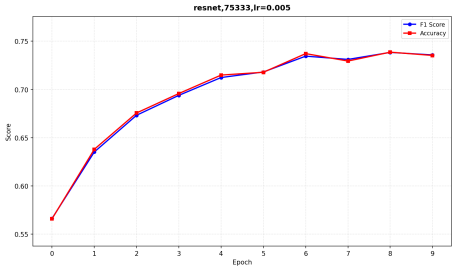

Figure 4: 75333,maxacc=0.7387,maxf1=0.7384

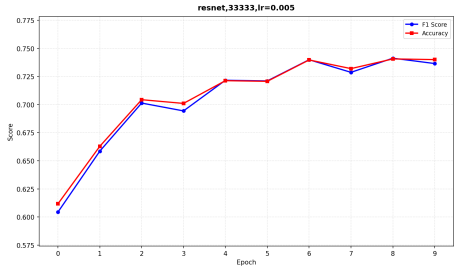

Figure 5: 33333,maxacc=0.7406,maxf1=0.7412

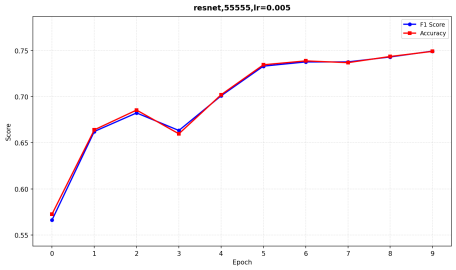

Figure 6: 55555,maxacc=0.7491,maxf1=0.7494

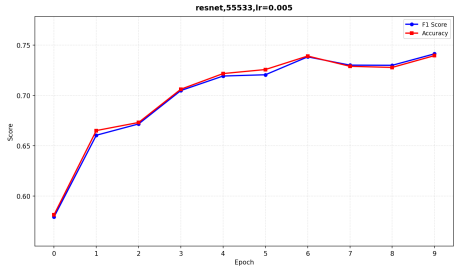

Figure 7: 55533,maxacc=0.7395,maxf1=0.7413

Figure 8: distinct kernel sizes

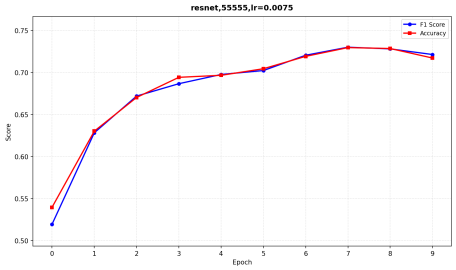

Figure 9: 0.0075,maxacc=0.7295,maxf1=0.7299

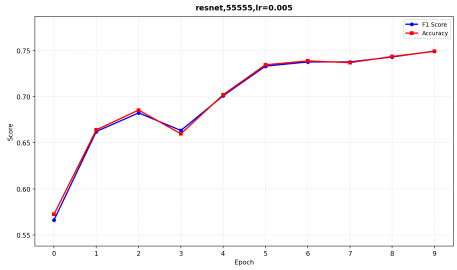

Figure 10: 0.005,maxacc=0.7491,maxf1=0.7494

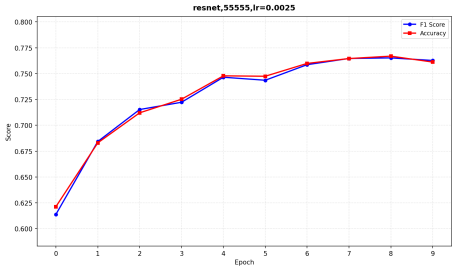

Figure 11: 0.0025,maxacc=0.7668,maxf1=0.7651

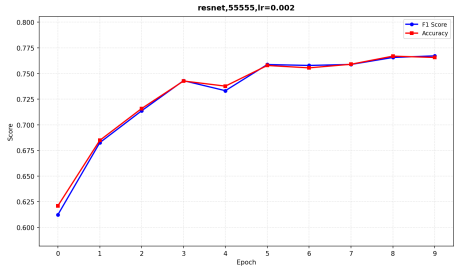

Figure 12: 0.002,maxacc=0.7669,maxf1=0.7672

Figure 13: different lr

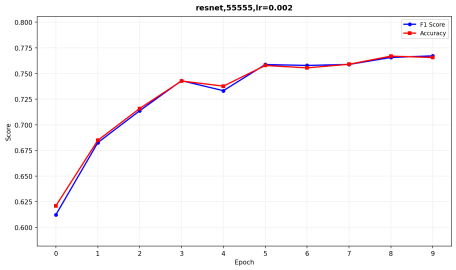

Figure 14: AdamW,maxacc=0.7669,maxf1=0.7672

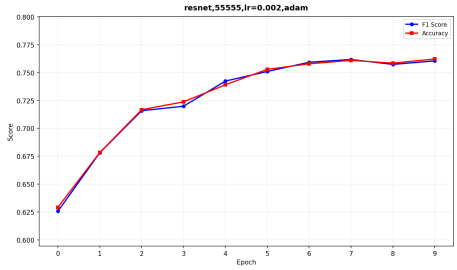

Figure 15: Adam,maxacc=0.7622,maxf1=0.7617

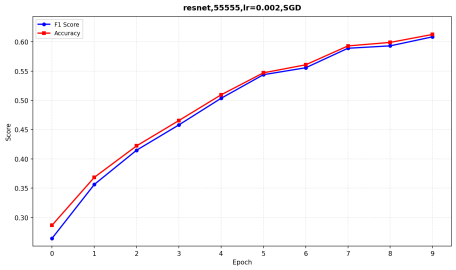

Figure 16: SGD,maxacc=0.6126,maxf1=0.6085

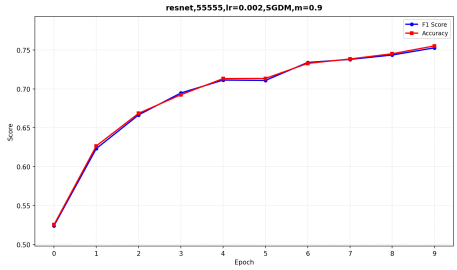

Figure 17: SGDM,maxacc=0.7553,maxf1=0.7528

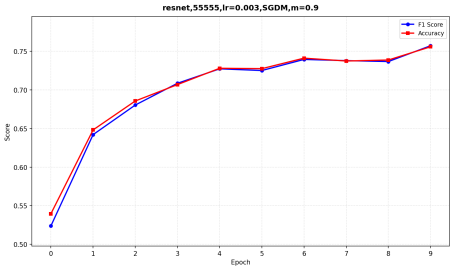

Figure 18: SGDM,lr=0.003,maxacc=0.7558,maxf1=0.7572

Figure 19: different optimizers, default lr=0.002

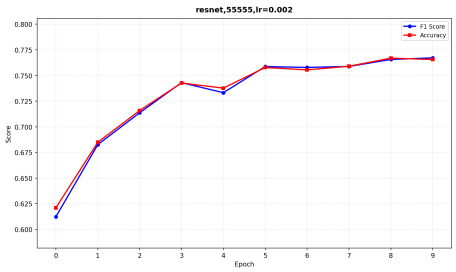

Figure 20: He uniform initialization,maxacc=0.7669,maxf1=0.7672

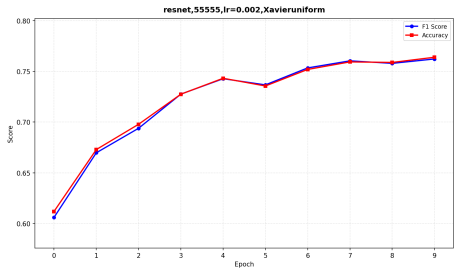

Figure 21: Xavier uniform initialization,maxacc=0.7639,maxf1=0.7622

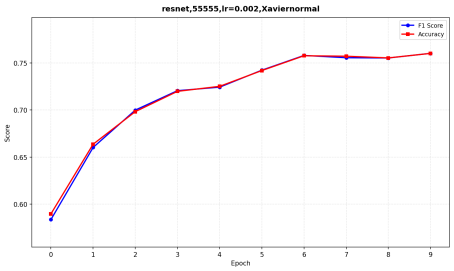

Figure 22: Xavier normal initialization,maxacc=0.7600,maxf1=0.7600

Figure 23: different initializing methods

model	time(s)
cnn	38.3
resnet	44.4
ViT	43.1

Table 1: time of doing 200000 inferences in my computer(with single GPU of 5070ti)

2 Experiments

2.1 performance of different models

2.1.1 accuracy

resnet is slightly better than cnn and much better than ViT. See 24.

Figure25 shows the top3 accuracy of resnet. It is apparently extremely high, proving its capability in certain high-fault-tolerance environment.

2.1.2 f1 score

Figure 26 and 27 show the f1 score of different types of targets. Cat and dog are very difficult figured out, possibly for their similar size may get the model confused. Bird is also difficult to classify as my kernel size is quite big, thus hindering some small features.

2.1.3 inference speed

1 shows the comparison of the inference speed of different models. Given resnet is much deeper, it's certainly reasonably slower, but surprisingly doesn't lag far behind the others. Maybe that's because most time is spent to deliver tensors to SM, as the test set is small-scale. My video memory occupation record supports this assumption: my GPU is just about 20% occupied by tensors, thus low efficiency.

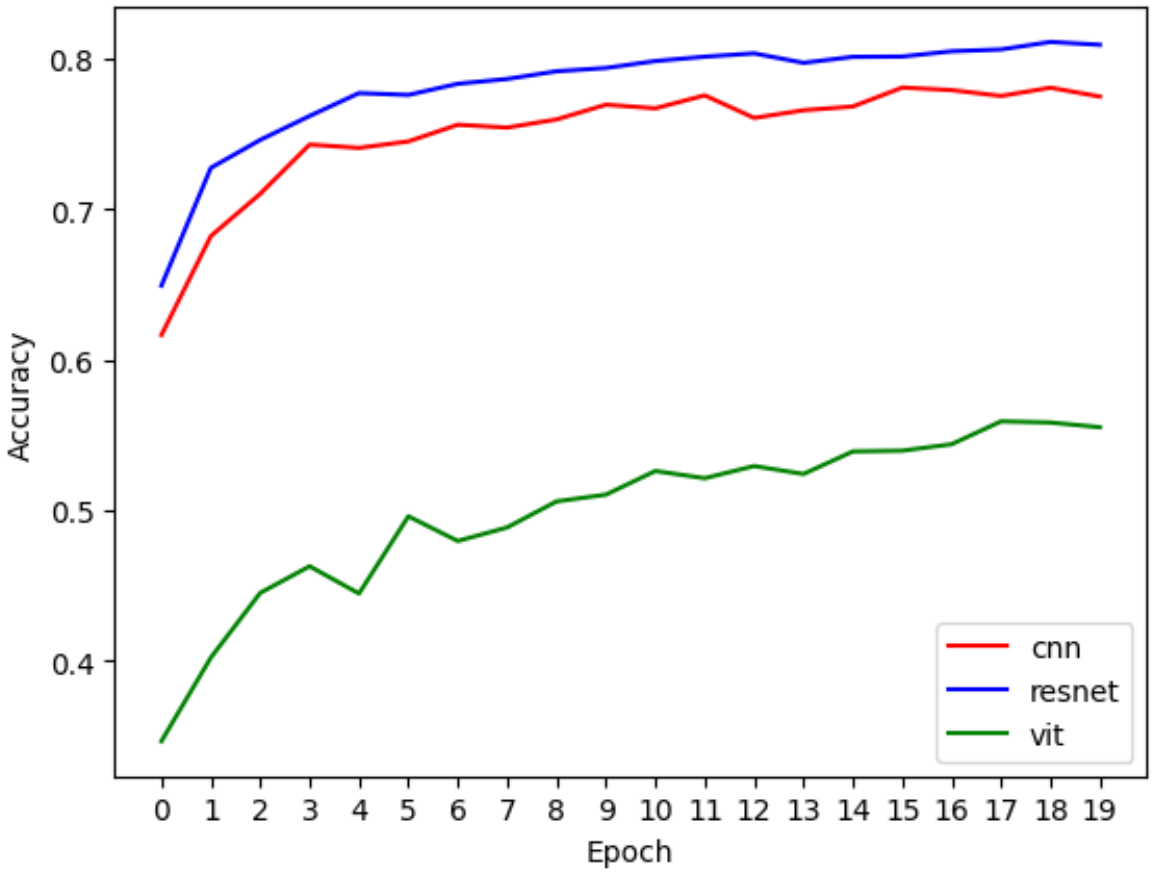

Figure 24: accuracy

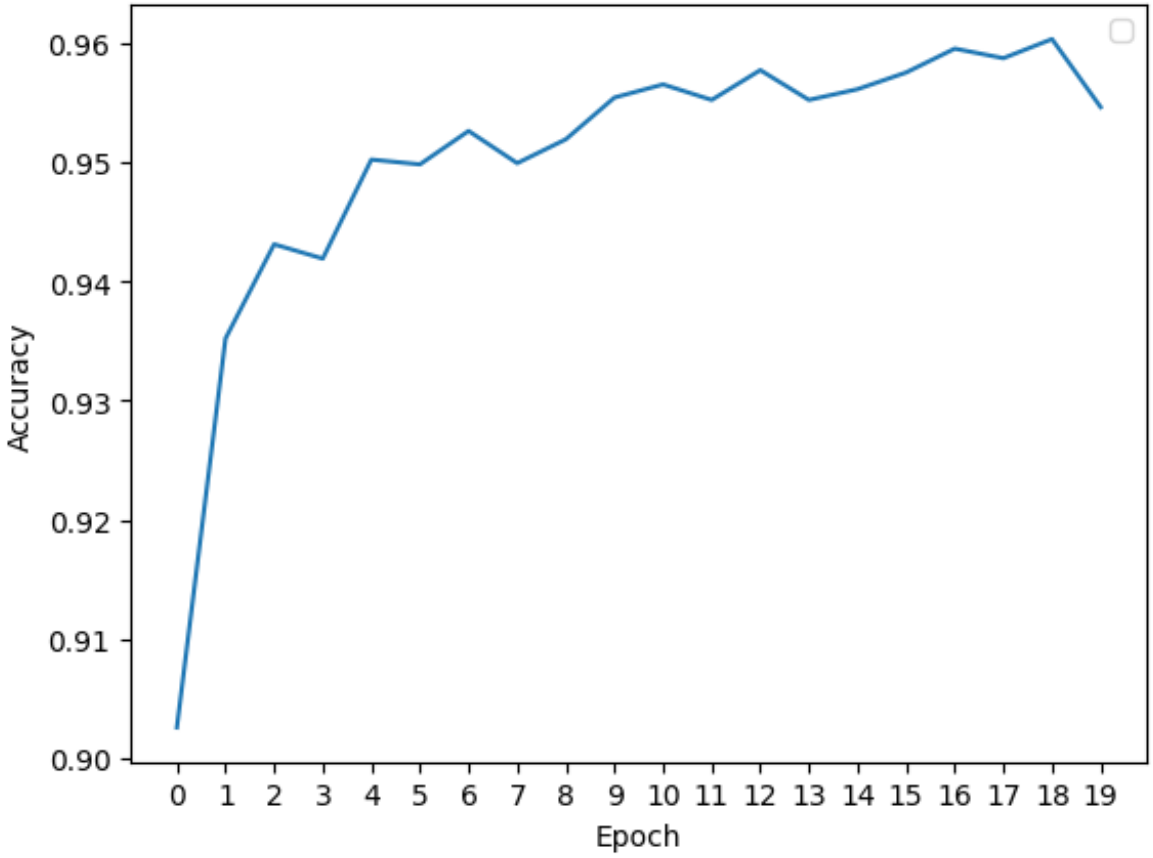

Figure 25: resnet top3 accuracy

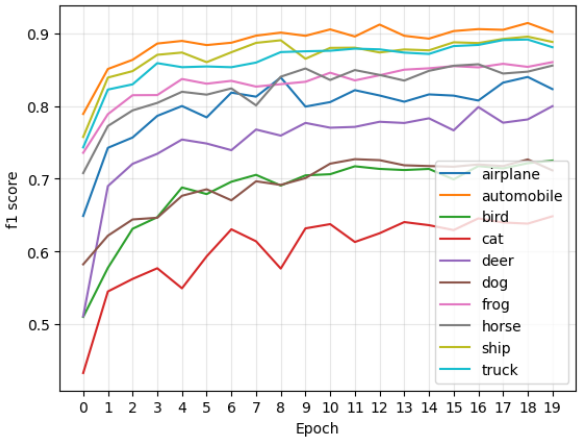

Figure 26: resnet f1 score

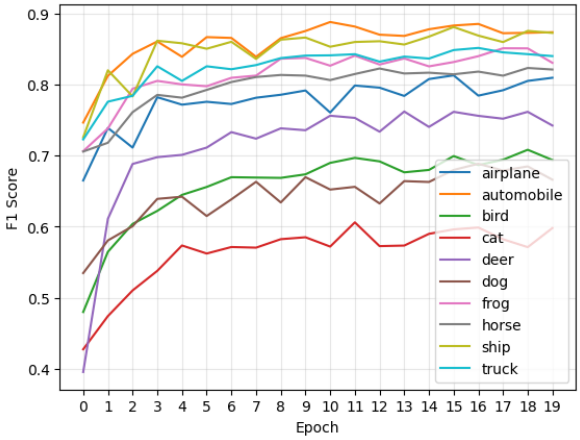

Figure 27: cnn f1 score

method	accuracy	f1 score
original resnet	0.8110	0.8095
resnet with layernorm and w/o batchnorm	0.5708	0.5701
resnet with dropout and w/o batchnorm	0.7558	0.7571

Table 2: attempt to replace batchnorm

2.2 other methods

2 shows my exploration of replacing batchnorm in resnet. The result shows batchnorm is definitely a key factor of training resnet.

3 Analysis for the result

There's one weird thing that ViT performs extremely disappointing, with just an accuracy of about 55% no matter how I tuned the hyperparameters (tuning process is recorded in my repository). I suspect the reason is that cifar-10 dataset is too small to converge my model (as my accuracy graph shows) and the number of my transformer layer is also too few, restricting its expressive potential. Additionally, I only conducted 20 epochs, obviously far from the needed epochs according to pytorch_cifar10 91%_cifar10-CSDN . All these factors result in its poor performance.

4 Generally speaking

In this report, I present a resnet that reached an accuracy of 81.10%, as well as some details during the training process. Codes are available in Litmeb/cifar10 .