Homework 4

600.482/682 Deep Learning

Spring 2019

Mou Zhang

Team Name: Mou Zhang

1. The first model: simple CNN with batchnorm, skip connection, data augmentation, regularization and different optimizers:

	batchnorm	skip connection	data augmentation	regularization	optimizers	dev accuracy	dev loss
simpleCNN	no	no	no	no	adam	0.396	2.673
simpleCNN	yes	no	no	no	adam	0.423	2.483
simpleCNN	no	yes	no	no	adam	0.428	2.529
simpleCNN	no	no	yes	no	adam	0.438	2.362
simpleCNN	no	no	no	yes	adam	0.374	2.532
simpleCNN	no	no	no	no	SGD	0.011	4.606
simpleCNN	yes	yes	yes	yes	adam	0.493	2.109

As we can see from the table, batchnorm, skip connnectin, and data augmentation all has a significant effect on this work.

(1) Batchnorm: batch normalization is supposed to reduce the inverse covariate shift. It transform data to ensure zero mean and unit variance to eliminate the vanishing gradient problem. In this problem, since it's a 5 layers CNN and vanishing gradient happens, we use the batch normalization to improve the performance.

(2) Skip connection: skip connection try to solve the same vanishing gradient problem in different way. It just set s shortcut connection between layers to avoid that problem. In this problem, since it's a 5 layers CNN and vanishing gradient happens, we use the batch normalization to improve the performance.

(3) Data augmentation: data augmentation is to solve the problem that the training set is too small and the distribution is not typical. In this problem, it is easy to figure out that randomcrop may work because the to-be-detect object may only exists in some part of the picture and the vertically flipped version of an object is common as well. So under this consideration. I use the random crop and horizontal flip to do data augmentation. It works well and increase accuracy by 5%

(4) Regularization: It is supposed to avoid overfitting, but it seems that it's not working well.

(5) Other optimizer(SGD): It's not working well, maybe because of the unchanged learning rate.

How to run:

(1)how to run the simple CNN with batchnorm, skip connection, data augmentation, regularization and optimizer Adam: just run the whole ipynb script in order (you may skip the google drive part).

(2)how to run code without batchnorm: in class SimpleCNN forwarding part, comment or delete 4 lines: x = self.bnm1(x), x = self.bnm2(x), x = self.bnm3(x), x = self.bnm4(x)

(3)how to run code without skip connection: in class SimpleCNN forwarding part, comment or delete 2 lines: x = x + self.sample_bnm1(self.downsample1(residual1)), x = x +

self.sample_bnm2(self.downsample2(residual2))

(4)how to run code without data augmentation: Do not run the function data_augmentation in this line: train_images, train_labels = data_augmentation(train_images, train_labels)

(5)how to run code without regularization: delete 'weight_decay=5e-4' in optimizer

(6)how to run code with different optimizers: change optimizer

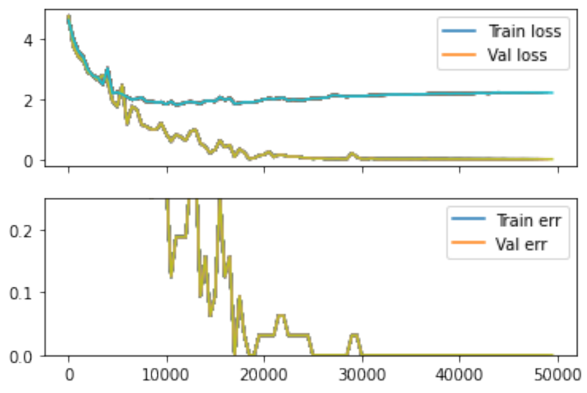
2. Resnet18 with data augmentation, batch normalization and lr_scheduler

	data augmentation	batch_norm	lr_scheduler	change learning rate manually	dev accuracy	dev loss
resnet18	no	no	no	no	0.504	2.143
resnet18	yes	no	no	no	0.588	2.057
resnet18	yes	yes	no	no	0.578	2.094
resnet18	yes	yes	ye	no	0.622	2.202

As we can see, the best strategies is to use data augmentation, batchnorm and lr_scheduler at the same time. We have discussed data augmentation and batchnorm above.

(1) Lr_scheduler means to change learning rate automatically. When the deep learning network is trained for a period of time, it will find the local minimum at the current step size and stabilized. At this time, only by changing the learning rate can the network find a smaller local minimum.

Below is the dev accuracy and loss for my best-performing model.



How to use:

(1)how to run the whole code: just run the whole ipynb script in order (you may skip the google drive part).

(2)without lr_scheduler: comment/delete the lr_scheduler part.

(3)without other part: the same as above.

3. Resnet18 with data augmentation and transfer learning

The abalation study table is shown as below

	transfer learning	data augmentation	dev accuracy	dev loss
resnet18	no	no	0.504	2.013
resnet18	yes	no	0.475	2.493
resnet18	yes	yes	0.514	2.389

The data augmentation is talked above.

Transfer Learning: As we know, transfer learning is to apply the network for one problem to another related problem. To reason is that the features extracted from original network is similar to the features we need in the new network. Since we are doing object classification on Cifar100, we can simply use other object classification model with transfer learning to do our job. In this transfer

learning, I used the pretrained resnet18(trained with minst) to do transfer learning. Since their features are similar, we have got a good result.
How to run:
(1)how to run the whole code: just run the whole ipynb script in order (you may skip the google drive part).
(2)run without transfer learning: set the resnet18(pretrained = False)
(3)without other part: the same as above.