HW 5

YI NIAN
MPCS Machine learning, University of Chicago

5-13

I discussed some of the problem with Hanqi Zhang.

Problem 1:

output image volume: 600*500*50
filter: 35*15*20
stride: 13

1:

Since the output depth = 50, then we will need 50 filters to generate 50, 600*500 image and stack them together.

2:

padding = 3
W2 = (W1 - KERNELW+2*PADDING)/S+1
H1 = (H2-1)*S+ KERNELH-2*PADDING = (600-1)*13+35-4 = 7818
W1 = (W2-1)*S+ KERNELW-2*PADDING = (500-1)*13+15-6 = 6496
depth = 20

Problem 2:

$DLoss/Df_{1,1,2}$ = $D_{11} * a_{112} + D_{12} * a_{122} + D_{21} * a_{212} + D_{22} * a_{222}$
$DLoss/Da_{2,2,2}$= $D_{11} * f_{222} + D_{12} * f_{212} + D_{21} * f_{122} + D_{22} * f_{112}$

Problem 4:

Model architecture:

MNIST:

1. For the design of my simple network, I use nn.sequential() as a container for a series of layers.

2. Input tensor size: (Batch size, channel,height,width)−− >(64,1,28,28)

Therefore, I applied two similar series of Conv-ReLU-Maxpool2d sequence to extract the feature map. The first Conv layer takes in 3 channels and output 16 channels.After first conv, we have tensor in shape (64,16,16,16) i.e:

Width and height after Conv:28-5+4+1 = 28

Width and height after maxpool:28-2/2 + 1 = 14

And after first sequence output will be in shape (64,16,14,14). An additional batchnorm layer was added to smooth the learning process.Similarly, the output of the second sequence will be (64,32,7,7).

3. For prediction of classes, I flatten each image map to be a single vector. And feed the output to a linear layer.The softmax process is actually done in the cross-entropy loss function.

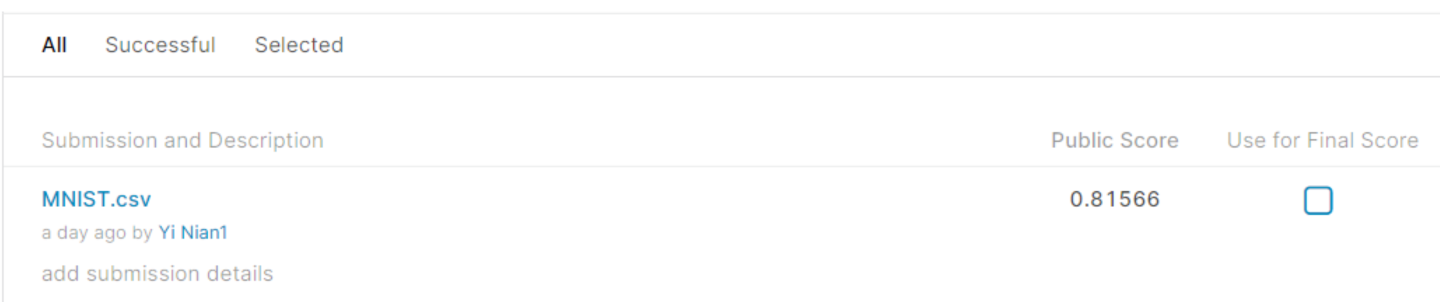
CIFAR100:

1. A network that similar to MNIST applied here will result in an accuracy of 38 percent on test data from Kaggle. So I decide to use a more complex network here.

2. After read the lecture notes from Stanford, I decide to use a similar architecture as VGG11(I add some batch norm layers and change some parameters) and it outperformed my simple network my roughly 7 percent.

Results:

MNIST Result:



1. 81 percent for two ”Conv2d-ReLu-Maxpooling” structure.

CIFAR Result

Submission and Description	Public Score	Use for Final Score
CIFAR2.csv 12 minutes ago by Yi Nian1 second submission	0.45200	☑
CIFAR.csv 20 hours ago by Yi Nian1 add submission details	0.38000	☐
CIFAR.csv 20 hours ago by Yi Nian1 initial submission YiNian	0.34100	☐

1. First submission, 34 percent, which is VGG11-like architechture, without batchnorm layers.
2. Second submission, 38 percent, which is just two ”Conv2d-ReLu-Maxpooling” structure.
3. Last submission, 45 percent, which is VGG11-like architecture with batchnorm layers.

Hyperparameters:

MNIST:

1. For the back-prop and parameter updating, I use Adam optimization for a more stable convergence since we only have 15 epoch.(also tried SGD, but it is worse than Adam here)

2. After adding the batch norm layer, the convergence speed become faster.

CIFAR:

1. Besides what I did for MNIST, I normalize the data at the beginning with the mean and standard deviation of CIFAR-100 images, which improve the accuracy for about 10 percent.

2. The learning rate of the optimizer is crucial to the training process, here I used 0.0005. The learning rate is too high will cause the network not converge well. While too low will cause the training process too slow.