Data Science Hw 6

工科海洋四 B07505015 梁瑞翔

State all the hyperparameters you need for training and how you tune them:

num_epochs: 30

train_batch_size: 64

eval_batch_size: 32

momentum: 0.95

lr(learining rate): 0.006

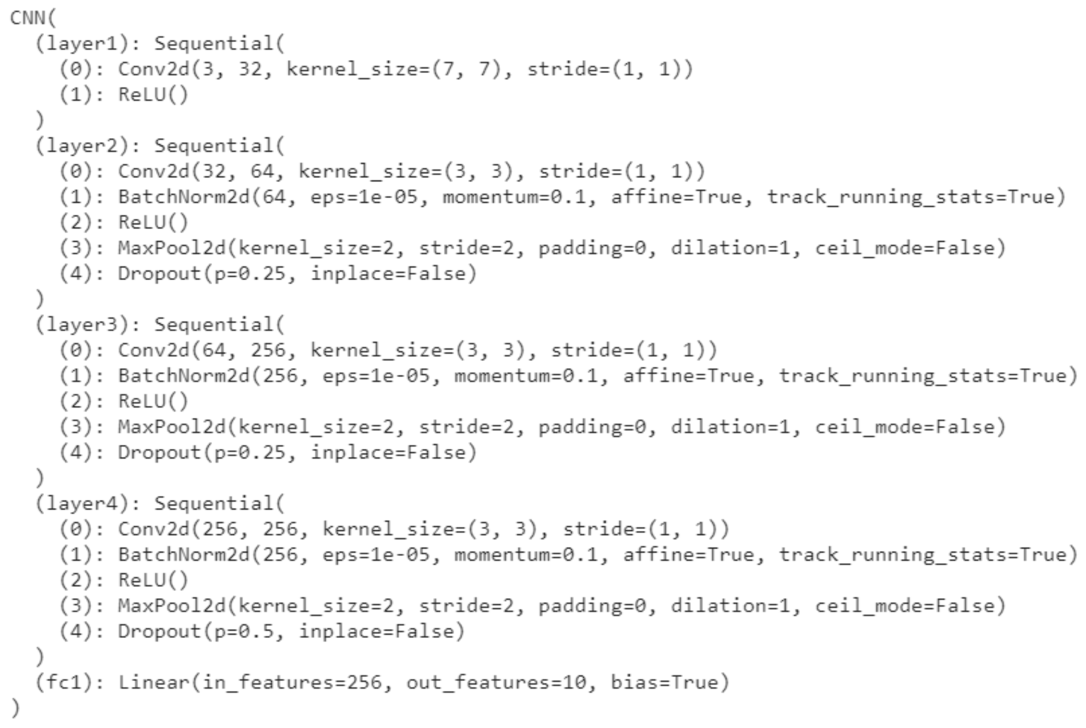
lr_gamma: 0.1

lr_decay_step: 20

weight_decay: 5e-4

My default value is the value in the sample code. According to my model, I need to set a larger epoch number to understand the trend of train loss. My default learning rate is 0.01, and I tune to next value by subtracting 0.002, which means the sequence of 0.01 0.008 0.006…, I found that when the value is at 0.006, the loss function seems to be fine. The values of momentum that I have tried was 0.9, 0.95 and 0.99, and the best result was obtained by setting it to 0.95.
I did not tune other default number because its accuracy is already over the strong baseline.

Show the structure of your best model:

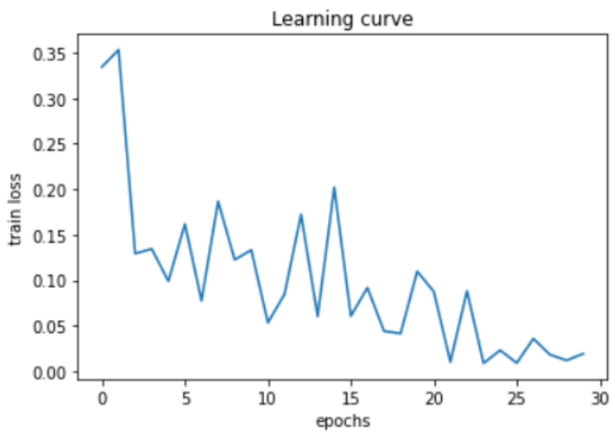


```
CNN(
  (layer1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(7, 7), stride=(1, 1))
    (1): ReLU()
  )
  (layer2): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (layer3): Sequential(
    (0): Conv2d(64, 256, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (layer4): Sequential(
    (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (fc1): Linear(in_features=256, out_features=10, bias=True)
)
```

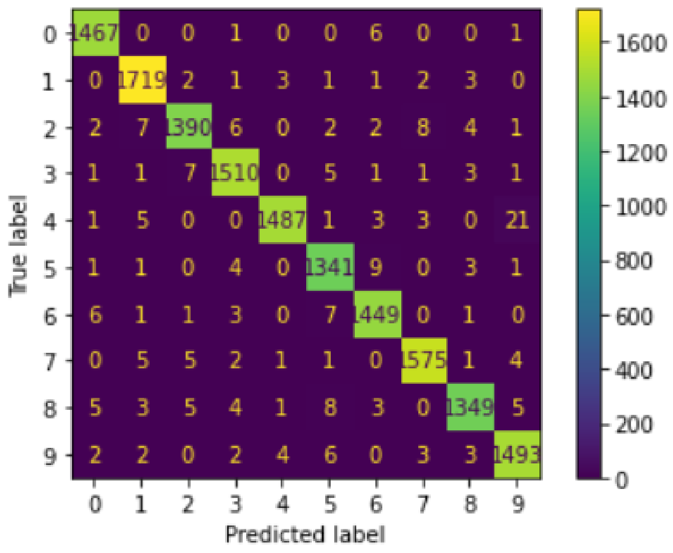
Explain the design of your model and what you've observed:

I create a four-level CNN model which acts better than a three-level model.
When I tried the three-level model with a 64x64 dense layer, the test accuracy can be around 98.x% or even higher, but the validation accuracy stops at 95.x%. I thought that was because the model is not that large to handle this kind of dataset, which contains thousands of images. The kernel size of the first layer I've tried was 5 and 7, and the result was close.
At the first epoch, the training accuracy was higher than the that of sample code. The dropout layer is important in CNN model because we can avoid the situation of overfitting. The parameters of that I tried were 0.5 first, and 0.35, 0.25.

Plot the learning curve during training:



Plot the confusion matrix for validation set, and briefly explain what you've observed:



	0	1	2	3	4	5	6	7	8	9
0	1467	0	0	1	0	0	6	0	0	1
1	0	1719	2	1	3	1	1	2	3	0
2	2	7	1390	6	0	2	2	8	4	1
3	1	1	7	1510	0	5	1	1	3	1
4	1	5	0	0	1487	1	3	3	0	21
5	1	1	0	4	0	1341	9	0	3	1
6	6	1	1	3	0	7	1449	0	1	0
7	0	5	5	2	1	1	0	1575	1	4
8	5	3	5	4	1	8	3	0	1349	5
9	2	2	0	2	4	6	0	3	3	1493

The most fault is between number 4 and 9, which may mean the feature of the two numbers are so close, and there may be some mistakes of the labels of the dataset.