NCTU Pattern Recognition, Homework 5

0886004 周芝妤

Coding (100%):

In this coding assignment, you need to implement the deep neural network by any deep learning framework, e.g. Pytorch, TensorFlow, or Keras, then train the DNN model by the Cifar-10 dataset and try to beat the baseline performance.

Download dataset HERE.

Please note that you should only train and evaluate your model on the provided dataset. **DO NOT** download the data from other resource**s.**

If you are a newbie in a deep learning framework, we recommend you learn **Keras** or **Pytorch**.

- Pytorch tutorial
- Keras tutorial
- TensorFlow tutorial

1. **(100%) Show your accuracy of your model on the provided test data by screenshot the results of your code and paste them on your report**

Evaluation:

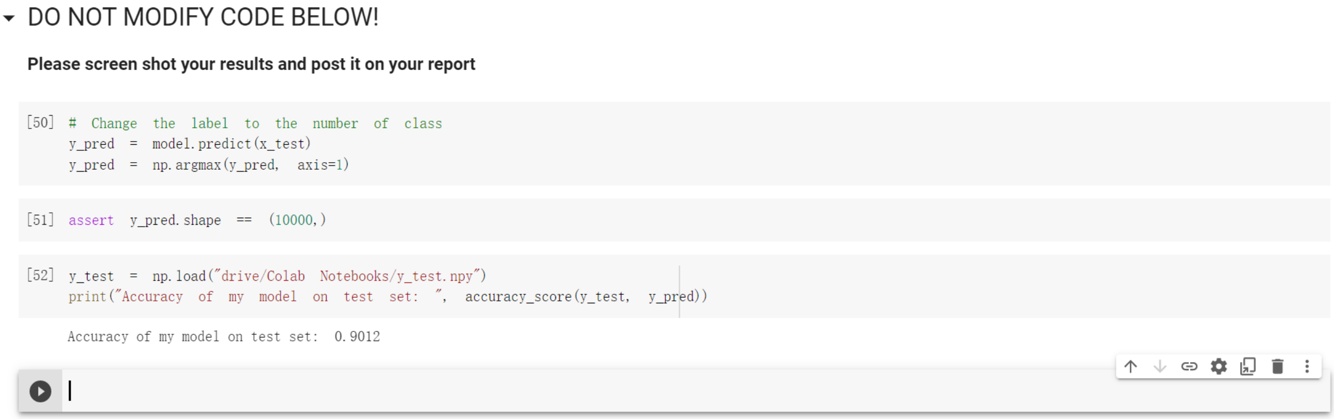
Accuracy	**Your scores**
acc >= 0.95	**100 points**
0.9 <= acc < 0.95	**90 points**
0.80 <= acc < 0.90	**80 points**
0.75 <= acc < 0.80	**70 points**
0.65 <= acc < 0.75	**60 points**
0.6 <= acc < 0.65	**50 points**
acc <0.6	**No points**

Note**:** Keyword to boost your model performance

1. Data augmentation
2. Hyperparameter searching for model structure (number of filers, number of convolution/dense layer) and optimizer (learning rate)
3. Regularization

Note: If your result is bad, check this tutorial first to debug your model

2. My performance:



DO NOT MODIFY CODE BELOW!

Please screen shot your results and post it on your report

```
[50] # Change the label to the number of class
     y_pred = model.predict(x_test)
     y_pred = np.argmax(y_pred, axis=1)
```

```
[51] assert y_pred.shape == (10000,)
```

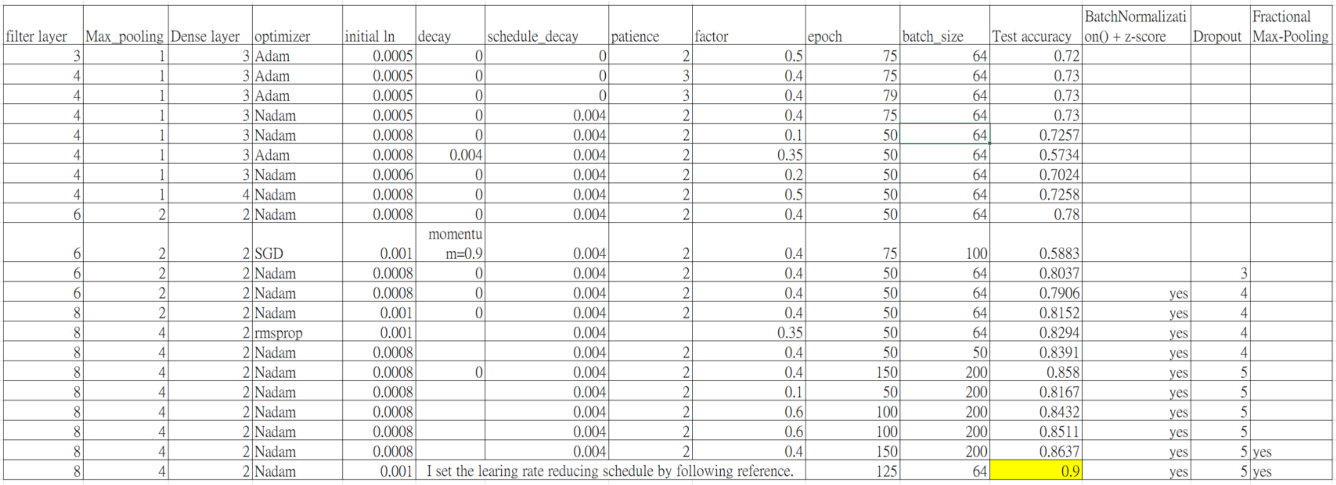
```
[52] y_test = np.load("drive/Colab Notebooks/y_test.npy")
     print("Accuracy of my model on test set: ", accuracy_score(y_test, y_pred))

     Accuracy of my model on test set: 0.9012
```

This is the screen shot from google Colab, I trained my model on it.
Also I added " y_pred = np.argmax(y_pred, axis=1) " after "y_pred = model.predict(x_test)" cause I needed to transfer the 10-dimension probability I got after prediction to the number of the most likely class. I didn't modify the other codes so I guess my change is permitted?

3. Implementation details and hyperparameters in my model:

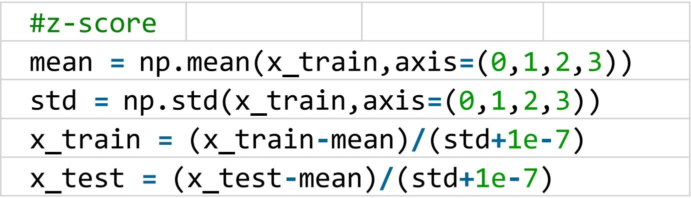
This table is my record of training with different parameters and in the last page I put the bigger version of this table.



filter layer	Max_pooling	Dense layer	optimizer	initial ln	decay	schedule_decay	patience	factor	epoch	batch_size	Test accuracy	BatchNormalization() + z-score	Dropout	Fractional Max-Pooling
3	1	3	Adam	0.0005	0	0	2	0.5	75	64	0.72			
4	1	3	Adam	0.0005	0	0	3	0.4	75	64	0.73			
4	1	3	Adam	0.0005	0	0	3	0.4	79	64	0.73			
4	1	3	Nadam	0.0005	0	0.004	2	0.4	75	64	0.73			
4	1	3	Nadam	0.0008	0	0.004	2	0.1	50	64	0.7257			
4	1	3	Adam	0.0008	0.004	0.004	2	0.35	50	64	0.5734			
4	1	3	Nadam	0.0006	0	0.004	2	0.2	50	64	0.7024			
4	1	4	Nadam	0.0008	0	0.004	2	0.5	50	64	0.7258			
6	2	2	Nadam	0.0008	0	0.004	2	0.4	50	64	0.78			
6	2	2	SGD	0.001	momentum=0.9	0.004	2	0.4	75	100	0.5883			
6	2	2	Nadam	0.0008	0	0.004	2	0.4	50	64	0.8037		3	
6	2	2	Nadam	0.0008	0	0.004	2	0.4	50	64	0.7906	yes	4	
8	2	2	Nadam	0.001	0	0.004	2	0.4	50	64	0.8152	yes	4	
8	4	2	rmsprop	0.001		0.004		0.35	50	64	0.8294	yes	4	
8	4	2	Nadam	0.0008		0.004	2	0.4	50	50	0.8391	yes	4	
8	4	2	Nadam	0.0008	0	0.004	2	0.4	150	200	0.858	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.1	50	200	0.8167	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.6	100	200	0.8432	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.6	100	200	0.8511	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.4	150	200	0.8637	yes	5	yes
8	4	2	Nadam	0.001	I set the learing rate reducing schedule by following reference.				125	64	0.9	yes	5	yes

Data preprocess

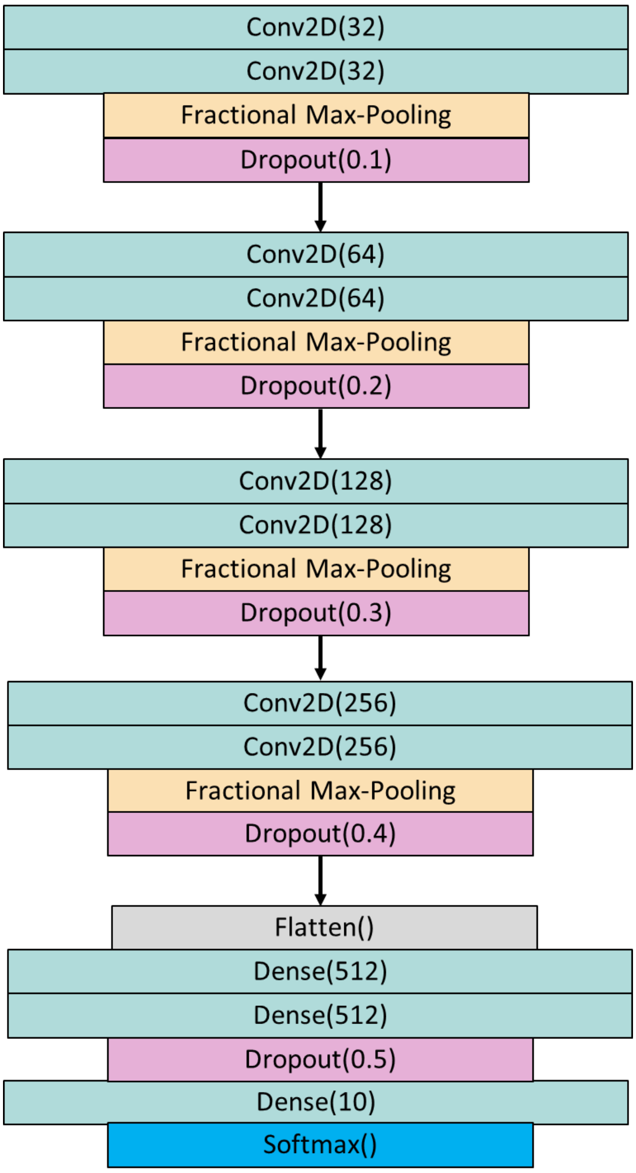
Instead of directly dived the data by 255, I used z-score to normalize data.
(The last 11 rounds I used z-score and it improved my accuracy from around 0.78 to 0.7906~0.81.)
This picture shows my code of implementation of z-score.



```
#z-score
mean  = np.mean(x_train,axis=(0,1,2,3))
std = np.std(x_train,axis=(0,1,2,3))
x_train = (x_train-mean)/(std+1e-7)
x_test  = (x_test-mean)/(std+1e-7)
```

Model Structure (referred to the B website**)**

This is the simplified graph of my structure, I tried to use different number of the filter layer, Max_pooling and sense layer. I got better performance when I used more layers of filter, and more pooling layers, but when I increased the dense layers, the accuracy didn't improve as I thought. I think filters and pooling contribute more then dense layer, if the model has enough convolution times and pooling, it only needs one or two dense layer to classify these data.



I used BatchNormalization() to normalize the activation of the previous layer at each batch as a regularizer. (Referred to the A website)

I also used Fractional Max-Pooling to replace the usual Max-Pooling Model, and it slight improved my performance from 0.8511 to 0.8637 in previous training. (Showed in the attached table)

The settings of learning rate and optimizer

I implement different optimizers, learning rate, decay and factor by ReduceLROnPlateau from keras.callbacks, and I get over 0.8 accuracy by using Nadam optimizer with 0.0008 for learning rate, 0.004 for schedule_decay, 0.4 for factor and patience= 2, but I couldn’t get over 0.9 although I tried different settings of other parameter like batch size, so in the end I try the setting in the reference A website and I got best performance (over 0.9), the code of decay schedule showed below:

```
def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 75:
        lrate = 0.0005
    if epoch > 100:
        lrate = 0.0003
    return lrate
```

By using this schedule, I forced the model to train by larger learning rate till over 75 epochs. I think the previous round when I use ReduceLROnPlateau to reduce learning rate by patience = 2, my model got stuck in the local minima, I guess if I change patience to larger number (like 5 or 8) may improve the performance in previous settings.

The settings of learning epoch and batch_size

I tried 50, 75, 100, 125, 150 epochs and 64, 100, 200 batch_size, and I get best accuracy by using epoch = 125 and batch_size = 64.

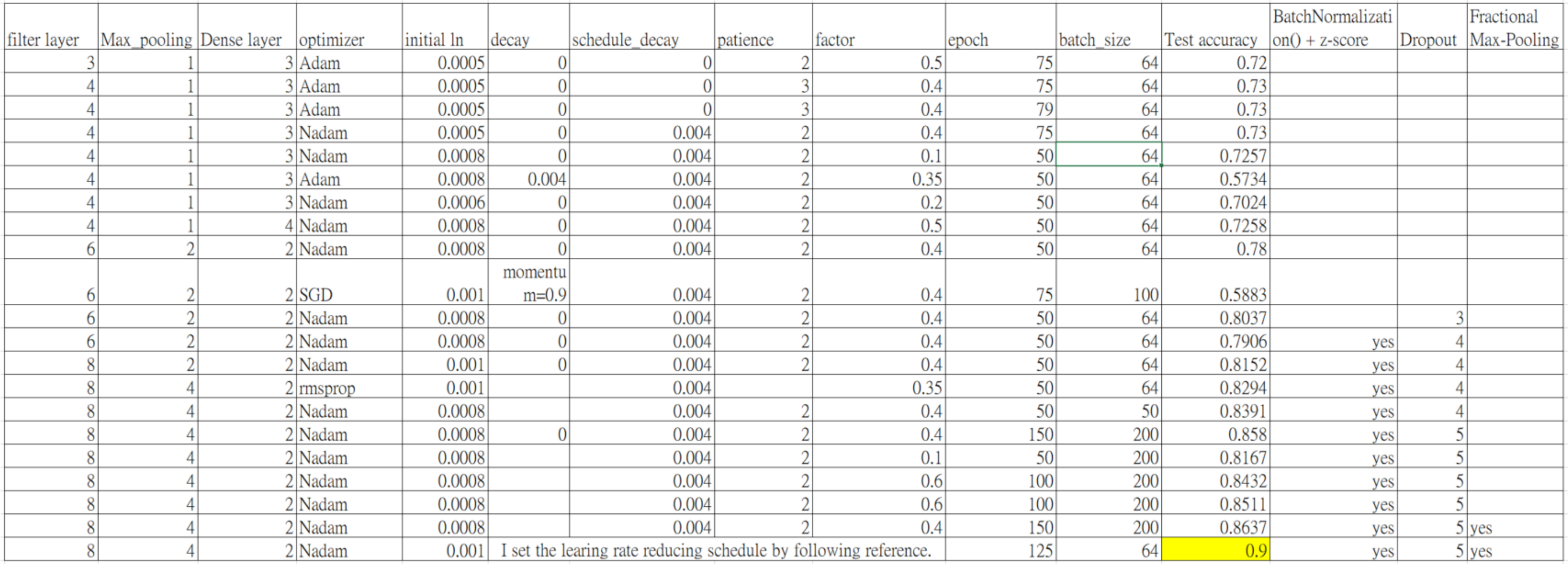
Data augmentation

I used ImageDataGenerator to generate data for training. The parameters I used is: rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode="nearest". I think to rotate image by 40 degree and shift it around 0.2 is a fine criteria cause I could recognize this image after the changes, and also flip the data horizontally. In our class we have the labels of “ automobile” and “ship”, I think these kind of image will be weird if I flip it vertically, so I only set horizontal_flip=True but not vertical.

I referred to this two websites for the structure and data normalization methods:

A. https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/?fbclid=IwAR3_p9HF8UTP4xKnPnzz_v6-OwkPoDZTdumLV_pmaBQWUW0g0EdUOIcv_1k

B. https://laplacetw.github.io/data-sci-vgg-cifar10/



filter layer	Max_pooling	Dense layer	optimizer	initial ln	decay	schedule_decay	patience	factor	epoch	batch_size	Test accuracy	BatchNormalization() + z-score	Dropout	Fractional Max-Pooling
3	1	3	Adam	0.0005	0	0	2	0.5	75	64	0.72			
4	1	3	Adam	0.0005	0	0	3	0.4	75	64	0.73			
4	1	3	Adam	0.0005	0	0	3	0.4	79	64	0.73			
4	1	3	Nadam	0.0005	0	0.004	2	0.4	75	64	0.73			
4	1	3	Nadam	0.0008	0	0.004	2	0.1	50	64	0.7257			
4	1	3	Adam	0.0008	0.004	0.004	2	0.35	50	64	0.5734			
4	1	3	Nadam	0.0006	0	0.004	2	0.2	50	64	0.7024			
4	1	4	Nadam	0.0008	0	0.004	2	0.5	50	64	0.7258			
6	2	2	Nadam	0.0008	0	0.004	2	0.4	50	64	0.78			
6	2	2	SGD	0.001	momentum=0.9	0.004	2	0.4	75	100	0.5883			
6	2	2	Nadam	0.0008	0	0.004	2	0.4	50	64	0.8037		3	
6	2	2	Nadam	0.0008	0	0.004	2	0.4	50	64	0.7906	yes	4	
8	2	2	Nadam	0.001	0	0.004	2	0.4	50	64	0.8152	yes	4	
8	4	2	rmsprop	0.001		0.004		0.35	50	64	0.8294	yes	4	
8	4	2	Nadam	0.0008		0.004	2	0.4	50	50	0.8391	yes	4	
8	4	2	Nadam	0.0008	0	0.004	2	0.4	150	200	0.858	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.1	50	200	0.8167	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.6	100	200	0.8432	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.6	100	200	0.8511	yes	5	
8	4	2	Nadam	0.0008		0.004	2	0.4	150	200	0.8637	yes	5	yes
8	4	2	Nadam	0.001	I set the learing rate reducing schedule by following reference.				125	64	0.9	yes	5	yes