

Assignment 1 CNN for image recognition

Name:王力

Student number:1901213145

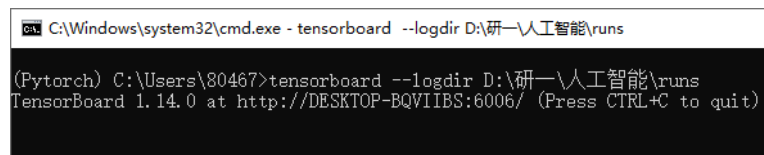
Date:2019.10.26

1.Run the Code

1.1. Run the code

Train: run train.py two times. New model for first time, continue train the model for second time. To save time, epoch is set to 1.

Visualize the learning progress: run visu.py generate runs folder. Then enter `tensorboard --logdir PATH` in the terminal, as Figure 1 shows. Copy link to browser, you can see the learning progress, include loss curve and accuracy curve.



```
C:\Windows\system32\cmd.exe - tensorboard --logdir D:\研一\人工智能\runs
(Python) C:\Users\80467>tensorboard --logdir D:\研一\人工智能\runs
TensorBoard 1.14.0 at http://DESKTOP-BQVIIBS:6006/ (Press CTRL+C to quit)
```

Figure 1 Instruction Input

Filter visualization: run visu_filter.py.

Feature mapping visualization: run visu_featuremap.py.

1.2. Directory structure and program function

File name	Function	Remarks
train.py	Start from scratch or continue to train a model The loss and accuracy of each epoch are stored in the datas.xls file for visualization.	To facilitate your test, I changed the model name to cifar_net_224_test.pth, and changed the epoch of training to one. After one epoch of training, you can run train.py again, and the model will continue to train.
network.py	Define network structure, forward propagation and evaluation function	
visu.py	Visualize loss and accuracy by tensorboardx	
visu_featuremap.py	Visualize feature map for conv1 and conv5_x	Take a sample randomly from the data set, input the model, and observe the output of conv1 and conv5_x
visu_filter.py	Visualize filters of conv1	conv1 has 64 filters, the

		size of each filter is 7x7, so I plot 64 subgraphs.
resnet_deconv.py	Definition of deconvolution neural network and forward propagation process	
deconv_demo.py	Perform deconvolution, display test image, conv1 reconstruction pattern and conv5-x reconstruction pattern	
cifar_net_224.pth	Trained model	
datas_224.xls	The loss and accuracy of each epoch are stored in the datas.xls file for visualization.	
cifar-10-batches-py	cifar-10 dataset	
runs	Folder for tensorboard visualization	

2. My own Resnet

2.1. Model Structure

The implementation of my own Resnet is in train.py program. Table 1 shows the structure of my own Resnet.

Table 1 Model Structure

Layer	Output Shape	Param #
Conv1	[-1, 64, 112, 112]	9536
Conv2-x	[-1, 64, 56, 56]	147968
Conv3-x	[-1, 128, 28, 28]	525568
Conv4-x	[-1, 256, 14, 14]	2099712
Conv5-x	[-1, 512, 7, 7]	8393728
FC	[-1, 10]	5130
Total params: 11,181,642		
Trainable params: 11,181,642		
Non-trainable params: 0		

2.2. Final Accuracy

The train and test accuracy curve are show in Figure 2 and Figure 3 respectively. The final training set accuracy is 1, and the test set accuracy is 83%. The loss curve is shown in Figure 4.

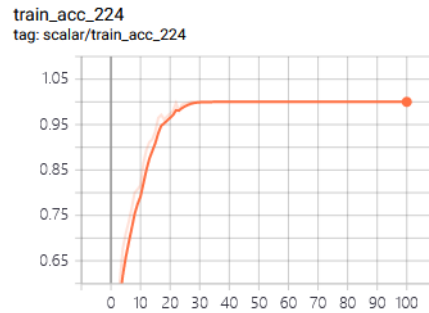


Figure 2 Train accuracy curve

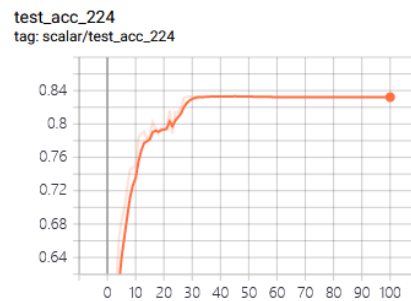


Figure 3 Test accuracy curve

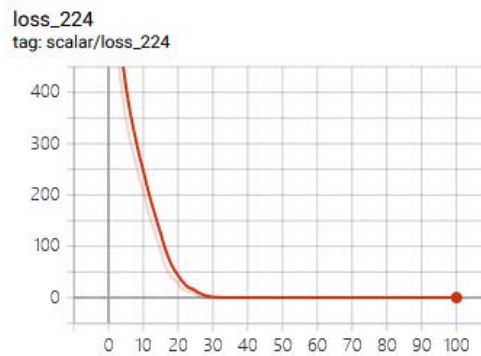


Figure 4 Loss curve

2.3. Filter visualization

The filter visualization is shown in Figure 5. Only the first channel of 64 filters is shown here.

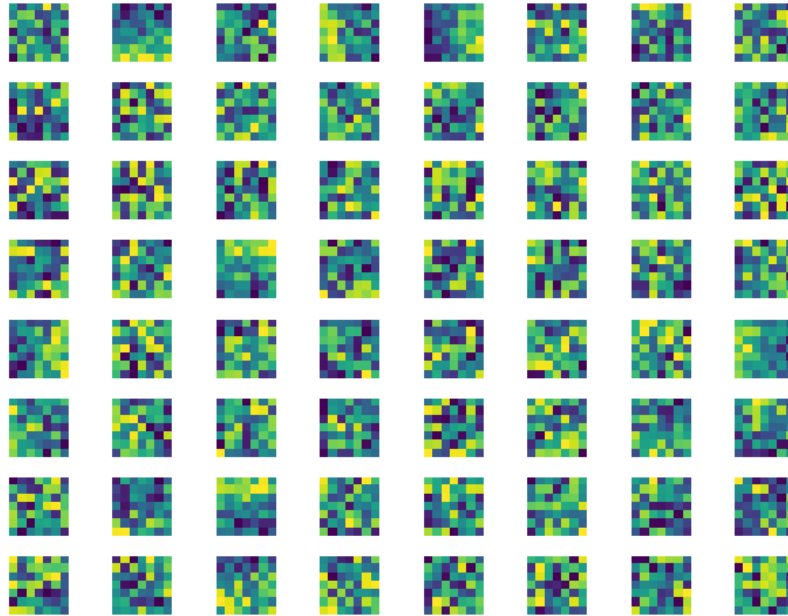


Figure 5 filter visualization

2.4. Feature mapping visualization

2.4.1. Corresponding image patches for each feature map

For show the corresponding image patches for each feature map, I randomly select a picture from the training set, input it into the model, and observe the output of conv1 and conv5_x layers respectively. Figure 6 shows the original image, Figure 7 and figure 8 show the output results of conv1 and conv5-x layers respectively.



Figure 6 Test sample

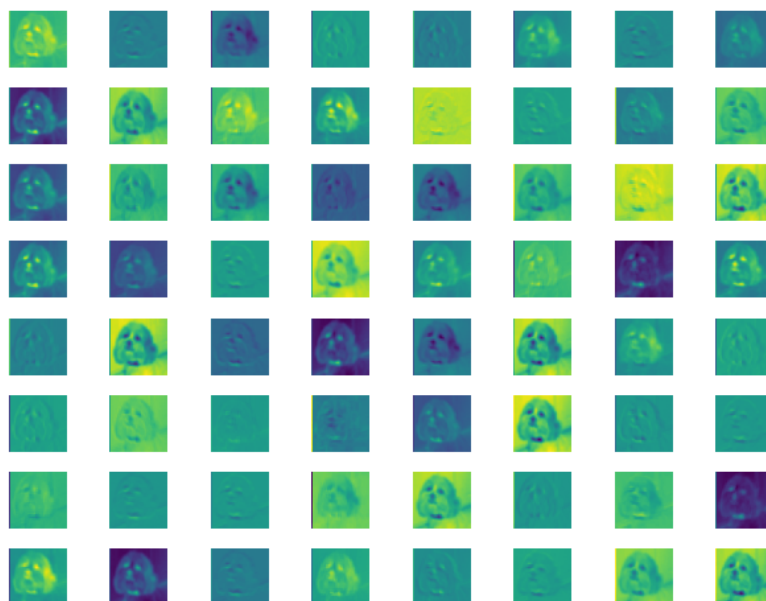


Figure 7 Feature map visualization of test sample(Conv1)

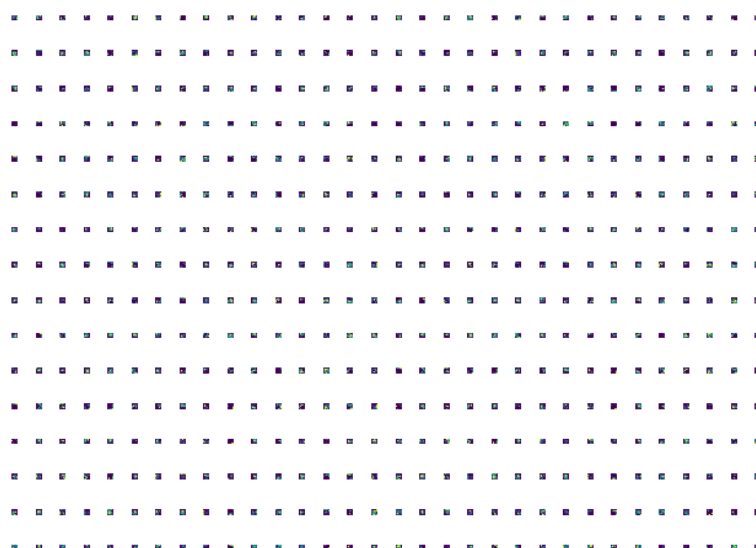


Figure 8 Feature map visualization of test sample(Conv5-x)

2.4.2. Reconstructed patterns

Figure 9 shows a sample of the test reconstruction pattern. Figure 10 and Figure 11 show the reconstruction patterns from the conv1 and conv5-x layers, respectively.

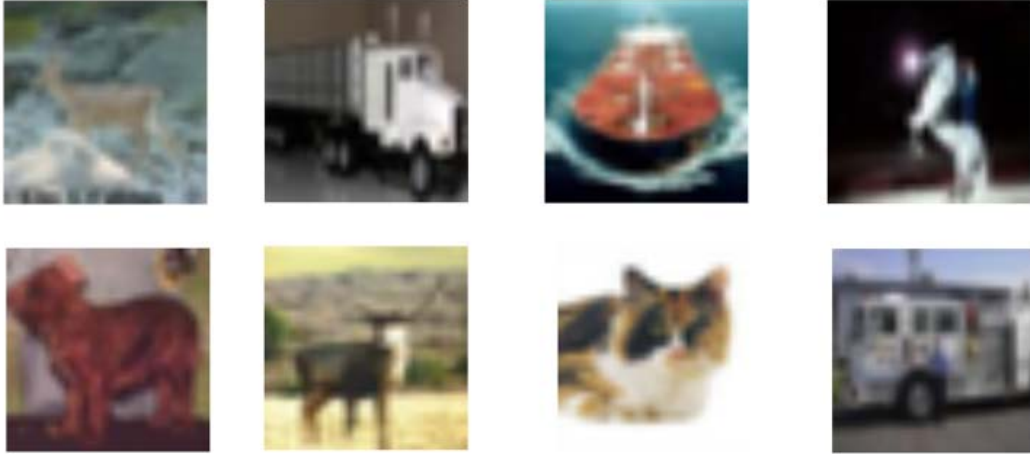


Figure 9 Test Sample

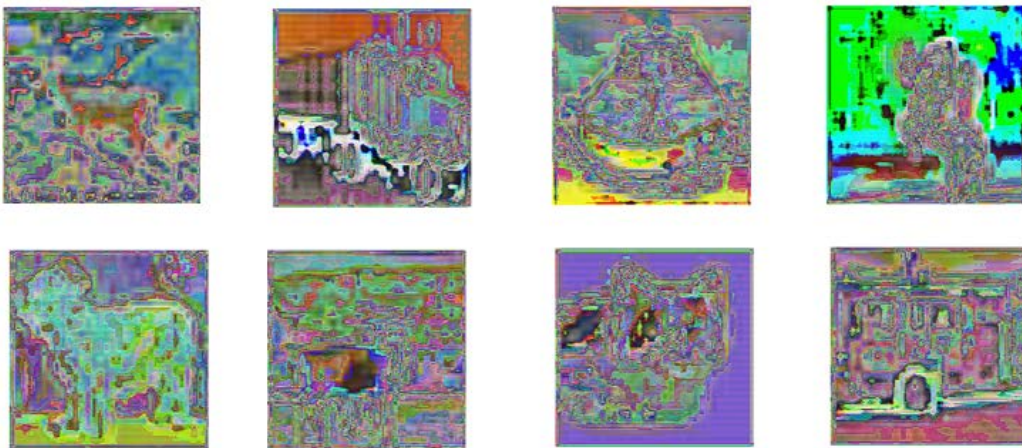


Figure 10 Reconstructed patterns from conv1 layer

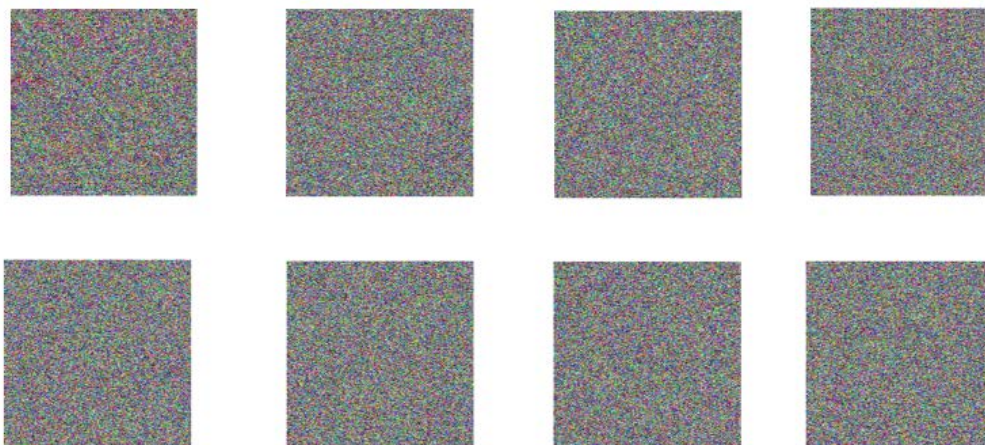


Figure 11 Reconstructed patterns from conv5-x layer

2.4.3. Batch normalization layer in reconstruction patterns

In [2], the operations of deconvolution layer, unpooling layer and unrelu layer are described, but the batch normalization layer is not mentioned. So we test the function of BN layer. The results with BN layer are shown in 2.4.2 Figure 12 show test sample, Figure 13 and Figure 14 show the reconstruction patterns of conv1 and conv5-x layer without BN layer respectively. Since conv1 does not pass through BN layer, the result in Figure 13 is similar to that in 2.4.2. **It can be seen from Figure 14 that the gradient disappears during the reconstruction mode without BN layer.**



Figure 12 Test Sample

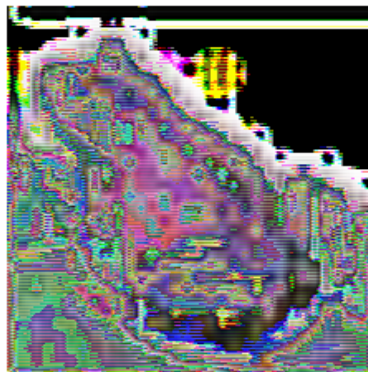


Figure 13 Reconstructed patterns from conv1 layer without BN layer



Figure 14 Reconstructed patterns from conv5-x layer without BN layer

3.Feedback

Q1: Time your spend for this assignment, i.e., how many hours?

A1: This assignment took me 48 hours, including coding, model training, model testing, parameter adjustment, visualization and report writing.

Q2: Comments for this course?

A2: This course has brought me a lot of benefits. I have never seen the teacher's ideas and model structures for problem solving before.

Q3: Comments for this assignment?

A3: First of all, thank you, TA. This assignment enables me to learn the usage of pytorch, the construction of Resnet, model learning, model evaluation and visualization. But this assignment document is not good enough.

Q4: Suggestion for the following lectures?

A4: Talk about some state-of-the-art technology, and the developing process. Focuses on idea rather than realization.

4.Reference

[1] https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html

[2] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In European conference on computer vision, pp.818–833.Springer, 2014.