Deep Learning Midterm Report

Min Jie
10109867
minjie@shanghaitech.edu.cn

Abstract

In this midterm project, I implement several different network structures and train on dataset CIFAR10. Also, given two subset of CIFAR 100 which called class1 and class2, I modify the network structure and parameters to achieve transfer learning task on the different datasets. Experiments based on the result of paper are presented below, details about setting and parameters value would show in the following parts.

1. Model implements

8 models in total (for assignment part1 and part2):

Model A, Model B, Model C, Strided-CNN-C, Conv-Pool-CNN-C, All-CNN-C, Transfer All-CNN-C class1, Transfer All-CNN-C class2

2. Experiments Analyze and Result Table

1. Experiment Environment

   Operating System : Linux 16.04

   Python version: Anaconda 3.6.5

   Pytorch verison: 0.4.1

2. Experiment Settings

   Overall, the weight initialazation strategy for all models are normalize on weights and constant 0.1 of bias in each convolution layers.

3. Dataset Division

   Train : Valid : Test = 48000 : 2000 : 10000

   TL task - Train : Valid : Test = 4000 : 1000 : 1000

4. Others

   Other details of my experiment setting, including model revision, weight initialization strategy, hyperparameters are wrote in the appendix 2 and 3 within each experiment.

Model	Paper Error	My Error
A	12.47%	14.63%
B	10.20%	14.10%
C	9.74%	13.40%
SCC	10.19%	13.70%
CPCC	9.31%	14.72%
ACC	9.08%	21.91%
TACC-class1	33.71%	33.60%
TACC-class2	33.71%	35.00%

Table 1. Results.

2.1. Analyze and Result Table

My experiments result is a little lower than the result of paper, I think it might due to insufficient training epoches and strategies.

However, I just analyze within my own experiments, the result correctly reflect the predicted tends of different structures in the paper.

For transfer learning task, I replace the last conv layers to amount of 100 to train 100 classes classifier on a different dataset CIFAR-100, the accuracy increase from 0 to the result in the paper, so it is reasonable.

3. Bonus

Explore the other transfer strategy on two given subset of CIFAR100 and reach your conclusion in report:

I tried to fix all weights and bias in very training process and just replace the last layer weight and bias to the origin fixed model in test part. Also, another strategy is to train a transfer function parameter beta and multiple in error part. Both of those two strategies benefit the solution sightly.

4. Appendix 1

4.1. Code structure

Two files (AllConv-v2.ipynb and cifar100.ipynb).

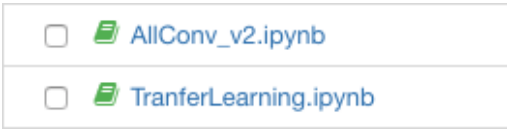
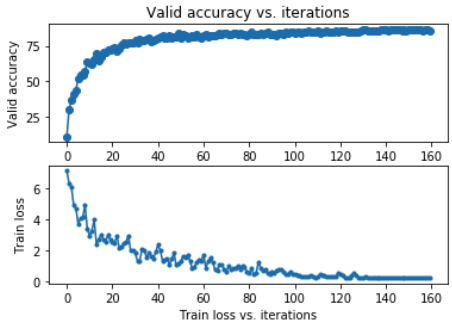


Figure 1. Code structure

The first file contains all codes and logs for assignment part 1: "AllConv Net on CIFAR-10 dataset". The second file contains all codes and logs for assignment part 2: "Transfer Learning on CIFAR-100 dataset".

5. Appendix 2 and 3

Model A Learning rate = 0.01, 25 epochs, decay each 5 epochs for decay rate = 0.8




```
test_accuracy(loader_test, best_model)
```

```
Checking accuracy on test set ====>
Got 8537 / 10000 correct (85.37)

85.37
```

Figure 2. Model A training curves and test accuracy

Model B Learning rate = 0.01, 25 epochs, decay each 5 epochs for decay rate = 0.85

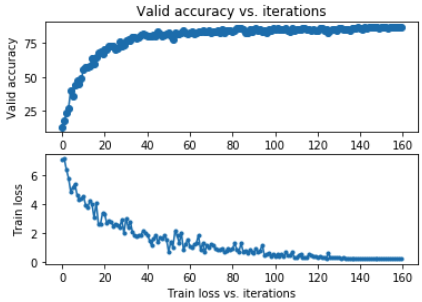
Model C Learning rate = 0.01, 25 epochs, decay each 5 epochs for decay rate = 0.85

Model SCC Learning rate = 0.01, 25 epochs, decay each 10 epochs for decay rate = 0.4

Model CPCC Learning rate = 0.01, 25 epochs, decay each 10 epochs for decay rate = 0.4

Model ACC Learning rate = 0.06, 60 epochs, decay each 15 epochs for decay rate = 0.5

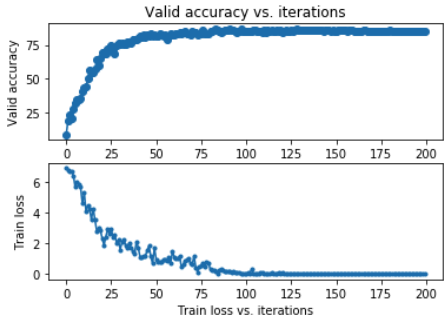
Model TACC-class1




```
test_accuracy(loader_test, best_model)
```

```
Checking accuracy on test set ====>
Got 8590 / 10000 correct (85.90)
```

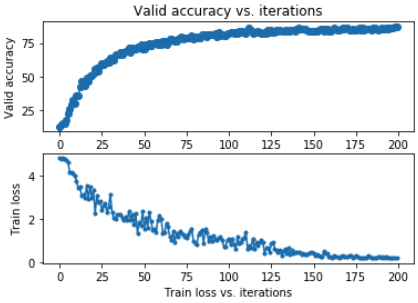
Figure 3. Model B training curves and test accuracy




```
test_accuracy(loader_test, modelC)
```

```
Checking accuracy on test set
Got 8555 / 10000 correct (85.55)
```

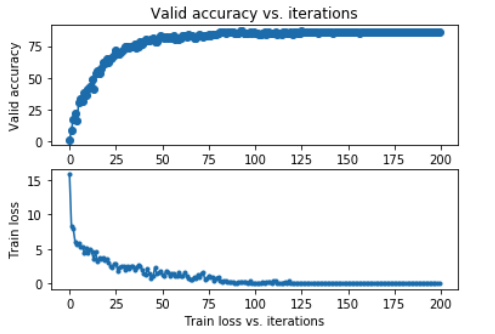
Figure 4. Model C training curves and test accuracy




```
test_accuracy(loader_val, modelCA)
```

```
Checking accuracy on test set ====>
Got 863 / 1000 correct (86.30)
```

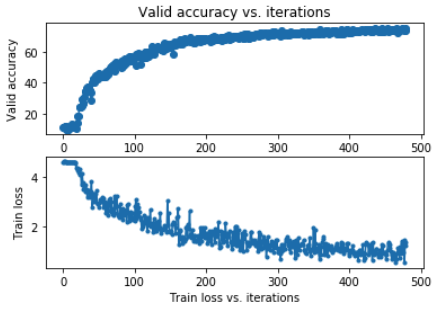
Figure 5. Model SCC training curves and test accuracy




```
test_accuracy(loader_test, modelCB)
```

```
Checking accuracy on test set
Got 8528 / 10000 correct (85.28)
```

Figure 6. Model CPCC training curves and test accuracy




```
test_accuracy(loader_test, modelCC)
```

```
Checking accuracy on test set ====>
Got 7799 / 10000 correct (77.99)
```

Figure 7. Model ACC training curves and test accuracy

Replace the last conv layer to 100 batches in order to fit 100 classes transfer learning class. Also add dropout layers and weights-decay to avoid overfit

Training:Validation:Test = 4000: 1000: 500

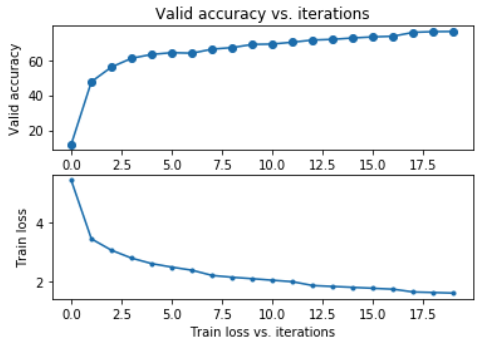
Learning rate = 0.001, 50 epochs, decay each 15 epochs for decay rate = 0.8

Model TACC-class2

Replace the last conv layer to 100 batches in order to fit 100 classes transfer learning class. Also add dropout layers and weights-decay to avoid overfit

Training:Validation:Test = 4000: 1000: 500

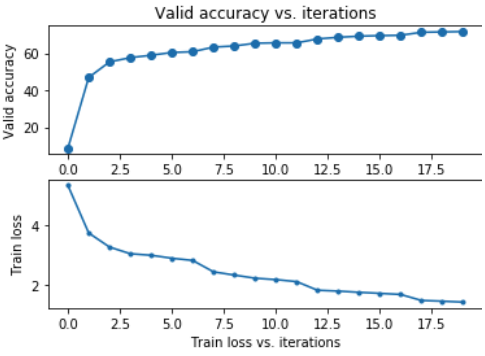
Learning rate = 0.001, 50 epochs, decay each 15 epochs for decay rate = 0.8




```
: test_accuracy(loader_test,modelCC)
```

```
Checking accuracy on test set
Got 664 / 1000 correct (66.40)
```

Figure 8. Model TACC-class1 training curves and test accuracy




```
test_accuracy(loader_test,modelCC)
```

```
Checking accuracy on test set
Got 650 / 1000 correct (65.00)
```

Figure 9. Model TACC-class2 training curves and test accuracy

5.1. References

1. Pytorch Document
2. Pytorch dataLoader source code
3. Previous homework assignment codes

```
Iteration 0, loss = 0.2172
Checking accuracy on validation set
Got 864 / 1000 correct (86.40)
Checking accuracy on test set ====>
Got 8529 / 10000 correct (85.29)
Iteration 100, loss = 0.2045
Checking accuracy on validation set
Got 864 / 1000 correct (86.40)
Checking accuracy on test set ====>
Got 8533 / 10000 correct (85.33)
Iteration 200, loss = 0.2027
Checking accuracy on validation set
Got 860 / 1000 correct (86.00)
Checking accuracy on test set ====>
Got 8531 / 10000 correct (85.31)
Iteration 300, loss = 0.2094
Checking accuracy on validation set
Got 864 / 1000 correct (86.40)
Checking accuracy on test set ====>
Got 8529 / 10000 correct (85.29)
Iteration 400, loss = 0.2031
Checking accuracy on validation set
Got 859 / 1000 correct (85.90)
Checking accuracy on test set ====>
Got 8533 / 10000 correct (85.33)
Iteration 500, loss = 0.2041
Checking accuracy on validation set
Got 864 / 1000 correct (86.40)
Checking accuracy on test set ====>
Got 8538 / 10000 correct (85.38)
Iteration 600, loss = 0.2077
Checking accuracy on validation set
Got 863 / 1000 correct (86.30)
Checking accuracy on test set ====>
Got 8540 / 10000 correct (85.40)
Iteration 700, loss = 0.2066
Checking accuracy on validation set
Got 858 / 1000 correct (85.80)
Checking accuracy on test set ====>
Got 8525 / 10000 correct (85.25)
```

Figure 10. Model A training logs

```
Iteration 0, loss = 0.2031
Checking accuracy on validation set
Got 861 / 1000 correct (86.10)
Checking accuracy on test set ====>
Got 8560 / 10000 correct (85.60)
Iteration 100, loss = 0.2022
Checking accuracy on validation set
Got 860 / 1000 correct (86.00)
Checking accuracy on test set ====>
Got 8578 / 10000 correct (85.78)
Iteration 200, loss = 0.2009
Checking accuracy on validation set
Got 863 / 1000 correct (86.30)
Checking accuracy on test set ====>
Got 8581 / 10000 correct (85.81)
Iteration 300, loss = 0.2009
Checking accuracy on validation set
Got 863 / 1000 correct (86.30)
Checking accuracy on test set ====>
Got 8581 / 10000 correct (85.81)
Iteration 400, loss = 0.2009
Checking accuracy on validation set
Got 863 / 1000 correct (86.30)
Checking accuracy on test set ====>
Got 8577 / 10000 correct (85.77)
Iteration 500, loss = 0.2005
Checking accuracy on validation set
Got 864 / 1000 correct (86.40)
Checking accuracy on test set ====>
Got 8596 / 10000 correct (85.96)
Iteration 600, loss = 0.2016
Checking accuracy on validation set
Got 863 / 1000 correct (86.30)
Checking accuracy on test set ====>
Got 8581 / 10000 correct (85.81)
Iteration 700, loss = 0.2025
Checking accuracy on validation set
Got 861 / 1000 correct (86.10)
Checking accuracy on test set ====>
Got 8582 / 10000 correct (85.82)
gotcha!
best_acc: 85.96
```

Figure 11. Model B training logs

```
Iteration 0, loss = 0.2004
Checking accuracy on validation set
Got 1722 / 2000 correct (86.10)
Checking accuracy on test set ====>
Got 8561 / 10000 correct (85.61)
Iteration 100, loss = 0.2005
Checking accuracy on validation set
Got 1723 / 2000 correct (86.15)
Checking accuracy on test set ====>
Got 8563 / 10000 correct (85.63)
Iteration 200, loss = 0.2007
Checking accuracy on validation set
Got 1722 / 2000 correct (86.10)
Checking accuracy on test set ====>
Got 8563 / 10000 correct (85.63)
Iteration 300, loss = 0.2001
Checking accuracy on validation set
Got 1722 / 2000 correct (86.10)
Checking accuracy on test set ====>
Got 8564 / 10000 correct (85.64)
Iteration 400, loss = 0.2017
Checking accuracy on validation set
Got 1722 / 2000 correct (86.10)
Checking accuracy on test set ====>
Got 8562 / 10000 correct (85.62)
Iteration 500, loss = 0.2010
Checking accuracy on validation set
Got 1724 / 2000 correct (86.20)
Checking accuracy on test set ====>
Got 8555 / 10000 correct (85.55)
Iteration 600, loss = 0.2006
Checking accuracy on validation set
Got 1722 / 2000 correct (86.10)
Checking accuracy on test set ====>
Got 8562 / 10000 correct (85.62)
Iteration 700, loss = 0.2004
Checking accuracy on validation set
Got 1723 / 2000 correct (86.15)
Checking accuracy on test set ====>
Got 8561 / 10000 correct (85.61)
no better
```

Figure 12. Model C training logs

```
Iteration 0, loss = 0.0016
Checking accuracy on validation set
Got 858 / 1000 correct (85.80)
Iteration 100, loss = 0.0097
Checking accuracy on validation set
Got 858 / 1000 correct (85.80)
Iteration 200, loss = 0.0031
Checking accuracy on validation set
Got 858 / 1000 correct (85.80)
Iteration 300, loss = 0.0017
Checking accuracy on validation set
Got 858 / 1000 correct (85.80)
Iteration 400, loss = 0.0006
Checking accuracy on validation set
Got 858 / 1000 correct (85.80)
Iteration 500, loss = 0.0014
Checking accuracy on validation set
Got 859 / 1000 correct (85.90)
Iteration 600, loss = 0.0041
Checking accuracy on validation set
Got 859 / 1000 correct (85.90)
Iteration 700, loss = 0.0009
Checking accuracy on validation set
Got 859 / 1000 correct (85.90)
```

Figure 13. Model CPCC training logs

```
Iteration 0, loss = 0.2242
Checking accuracy on validation set
Got 852 / 1000 correct (85.20)
Checking accuracy on test set ====>
Got 8318 / 10000 correct (83.18)
Iteration 100, loss = 0.2528
Checking accuracy on validation set
Got 860 / 1000 correct (86.00)
Checking accuracy on test set ====>
Got 8396 / 10000 correct (83.96)
Iteration 200, loss = 0.2131
Checking accuracy on validation set
Got 856 / 1000 correct (85.60)
Checking accuracy on test set ====>
Got 8405 / 10000 correct (84.05)
Iteration 300, loss = 0.2256
Checking accuracy on validation set
Got 865 / 1000 correct (86.50)
Checking accuracy on test set ====>
Got 8388 / 10000 correct (83.88)
Iteration 400, loss = 0.2175
Checking accuracy on validation set
Got 865 / 1000 correct (86.50)
Checking accuracy on test set ====>
Got 8395 / 10000 correct (83.95)
Iteration 500, loss = 0.2030
Checking accuracy on validation set
Got 862 / 1000 correct (86.20)
Checking accuracy on test set ====>
Got 8400 / 10000 correct (84.00)
Iteration 600, loss = 0.2145
Checking accuracy on validation set
Got 872 / 1000 correct (87.20)
Checking accuracy on test set ====>
Got 8401 / 10000 correct (84.01)
Iteration 700, loss = 0.2081
Checking accuracy on validation set
Got 868 / 1000 correct (86.80)
Checking accuracy on test set ====>
Got 8400 / 10000 correct (84.00)
gotcha!
best_acc: 84.05
```

Figure 14. Model SCC training logs

```
====> rounds: 29
Epoch====>0
Iteration 0, loss = 1.6041
Checking accuracy on validation set
Got 772 / 1000 correct (77.20)
Checking accuracy on test set ====>
Got 7791 / 10000 correct (77.91)
Iteration 100, loss = 1.4896
Checking accuracy on validation set
Got 770 / 1000 correct (77.00)
Checking accuracy on test set ====>
Got 7789 / 10000 correct (77.89)
Iteration 200, loss = 1.3269
Checking accuracy on validation set
Got 771 / 1000 correct (77.10)
Checking accuracy on test set ====>
Got 7797 / 10000 correct (77.97)
Iteration 300, loss = 1.1544
Checking accuracy on validation set
Got 771 / 1000 correct (77.10)
Checking accuracy on test set ====>
Got 7794 / 10000 correct (77.94)
Iteration 400, loss = 1.0255
Checking accuracy on validation set
Got 770 / 1000 correct (77.00)
Checking accuracy on test set ====>
Got 7787 / 10000 correct (77.87)
Iteration 500, loss = 1.2553
Checking accuracy on validation set
Got 771 / 1000 correct (77.10)
Checking accuracy on test set ====>
Got 7790 / 10000 correct (77.90)
Iteration 600, loss = 0.9308
Checking accuracy on validation set
Got 771 / 1000 correct (77.10)
Checking accuracy on test set ====>
Got 7788 / 10000 correct (77.88)
Iteration 700, loss = 0.9324
Checking accuracy on validation set
Got 772 / 1000 correct (77.20)
Checking accuracy on test set ====>
Got 7794 / 10000 correct (77.94)
no better
best_acc: 78.09
```

Figure 15. Model ACC training logs