The All Convolutional Net for CIFAR10 and Transfer Learning on CIFAR100

Yingying Ma
SIST
mayy1@shanghaitech.edu.cn

Abstract

In the paper, we implement All Convolutional Network in and other 5 network sturctures on which it based. We compare performance of these six models and found out ConvPool-CNN-C performs better than other models instead of All Convolutional Net as expected.

Besides, we finish transfer learning on subset class1 and class 2 of CIFAR-100 datasets.

1. Introduction

Most modern convolutional neural networks (CNNs) used for object recognition are built using the same principles: Alternating convolution and max-pooling layers followed by a small number of fully connected layers. In Paper [1], the authors discovered that max-pooling could simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks. For instance, max-pooling with stride 2 can be replaced by another convolutional layer with stride 2.

In the paper, we construct three basic CNN models and three advaced models, in which Strided-CNN-C and ALL-CNN-C replace max-pooling layer with convolution layer with stride 2. We compare performance of these six models and found out ConvPool-CNN-C is the best model to classify CIFAR10 datasets.

2. Model description

2.1. Basic models

Model A is a traditional model for CNN with convolutional layer following by max-pooling layer except for the 1ast two convolutional layers with the 1×1 kernel, which is used to replace the fully convolution layer. If the image area covered by units in the topmost convolutional layer covers a portion of the image large enough to recognize its content.

2.2. Advanced models

The Advanced models are shown in 2 The higher layers are the same as in Table 1. ConvPool-CNN-C does not use convolution layer with bigger stride. Strided-CNN-C and ALL-CNN-C is new models brought up by [1]. ALL-CNN-C add another convolution layer compared with Strided-CNN-C.

Model

A	B	C
Input 32×32 RGB image		
5×5 conv. 96 ReLU	5×5 conv. 96 ReLU 1×1 conv. 96 ReLU	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU
3×3 max-pooling stride 2		
5×5 conv. 192 ReLU	5×5 conv. 192 ReLU 1×1 conv. 192 ReLU	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU
3×3 max-pooling stride 2		
3×3 conv. 192 ReLU		
1×1 conv. 192 ReLU		
1×1 conv. 10 ReLU		
global averaging over 6×6 spatial dimensions		
10 or 100-way softmax		

Figure 1. Three basic models

Model

Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
Input 32×32 RGB image		
3×3 conv. 96 ReLU 3×3 conv. 96 ReLU with stride $r = 2$	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU 3×3 conv. 96 ReLU	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU
	3×3 max-pooling stride 2	3×3 conv. 96 ReLU with stride $r = 2$
3×3 conv. 192 ReLU 3×3 conv. 192 ReLU with stride $r = 2$	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU 3×3 conv. 192 ReLU	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU
	3×3 max-pooling stride 2	3×3 conv. 192 ReLU with stride $r = 2$

$\vdots$

Figure 2. Three advanced models

3. Transfer learning

In this part we do classifition task on subset class1 and class2 of CIFAR-100 datasets. We save the best ALL-CNN-C model and try to use it as a feature extractor. Replace and retrain the final FC classifier(which is the 1×1 conv layer), while fine-tuning the parameters of other layers.

4. Experiments

4.1. Experiment environment

Our sever is based on Ubuntu 16.04 and has 2 GTX 1080Ti GPU. Pytorch is used for training deep learning network. In order to speed up the training progress, we installed cuda9.0. Other than that, we install some packages like PIL for image loading.

4.2. Experiment setup

All networks were trained using stochastic gradient descent with fixed momentum of 0.9.

In classification task, we set original learning rate to be 0.01, weight decay to be 0.001 and train the network for 350 epochs. In epoch [200, 250, 300], we multiply 0.1 to the learning rate. The dropout layer is added and the probabilities were 20% for dropping out inputs and 50% otherwise.

In Transfer learning task, first we construct our own class of CIFAR100 to load the given data and transform the data into dataloader in torch. Then we load the best model parameters and set the weight of the last layer to be random and the bias to be 0. We use a list of learning rate of [0.0025, 0.001, 0.0005] to re-train the net. The total epoch number is 100, weight decay is also set 0.001.

4.3. Classification results

In order that we could compare the performance of each network, we list the validation accuracy of each model in the table1. As we can see ALL-CNN-C is the second best model. ConvPool-CNN-C with convolutional max-pooling layer performs better than ALL-CNN-C.

During our training, the three advanced model is easy to stuck in the local-minimal and hard to converge compared with the three basic cnn models, especially ALL-CNN-C.

Model	Accuracy
Model A	0.864
Model B	0.844
Model C	0.869
Strided-CNN-C	0.864
ConvPool-CNN-C	0.891
ALL-CNN-C	0.869

Table 1. Accuracy Comparison

4.4. Transfer learning

We could see that transfer learning is much more better than random initializing the parameters and less likely to stuck in the local-minimal point.

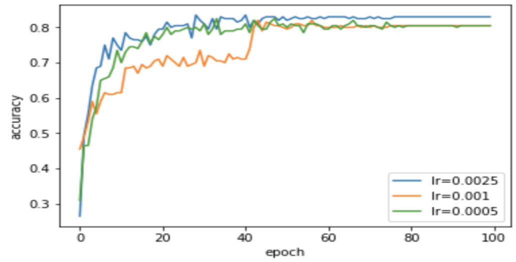

Figure 3. Transfer learning Class1 Accuracy

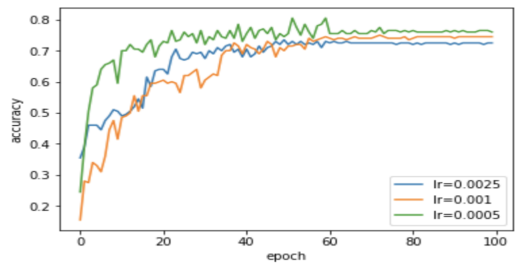

Figure 4. Transfer learning Class2 Accuracy

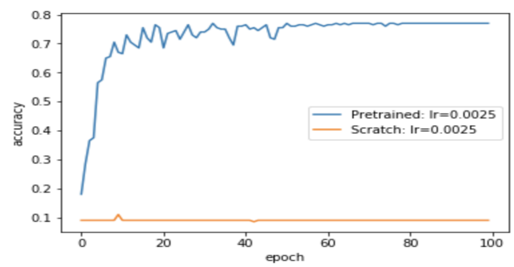

Figure 5. Class1 Transfer vs. Scratch learning Accuracy

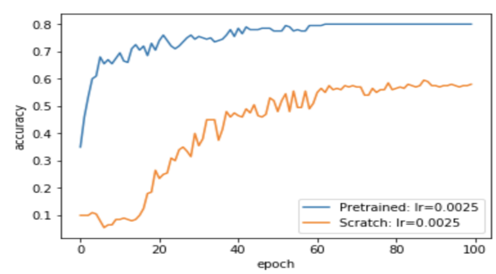

Figure 6. Class2 Transfer vs. Scratch learning Accuracy

5. Conclusion

In the paper, we construct the models that [1] metioned and find that the new model is not robust and doest not perform better than ConvPool-CNN-C with conventional max-pooling layer. So the function of CNN with higher stride need need to be discussed later. Transfer learning is a great trick to initialize network parameters. The network with transfer learning is more likely to converge and converge faster.

6. Appendix

6.1. Appendix 1

Each ipynb file in the code structure construct one model. Each file consists of basic load data, class Model, train function. The flow is shown as following:

Figure 7. Code Structure

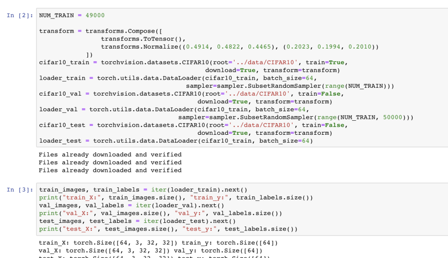

Figure 8. 1_Load Data

Figure 9. 2_Class model

Figure 10. 3_Utility functions

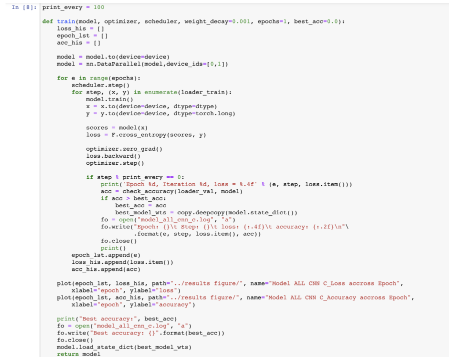

Figure 11. 4_train function

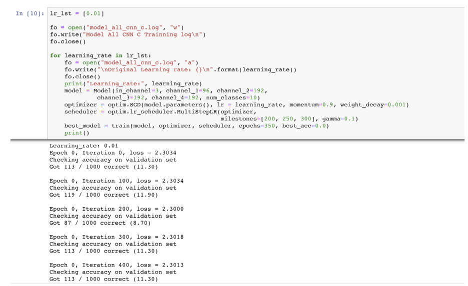

Figure 12. 5_train model

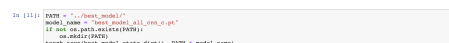

Figure 13. 6_save the best model

6.2. Appendix 2

Appendix 2 includes the screen shot of the evaluation and training log. The last line is the validation accurate for each model From the log file we could see that all model converge. But during our training, the three advanced model is hard to converge compared with the three basic cnn models.

```
Epoch: 348  Step: 0    loss: 0.0118  accuracy: 0.85
Epoch: 348  Step: 100  loss: 0.0175  accuracy: 0.85
Epoch: 348  Step: 200  loss: 0.0151  accuracy: 0.85
Epoch: 348  Step: 300  loss: 0.0244  accuracy: 0.85
Epoch: 348  Step: 400  loss: 0.0217  accuracy: 0.85
Epoch: 348  Step: 500  loss: 0.0229  accuracy: 0.85
Epoch: 348  Step: 600  loss: 0.0113  accuracy: 0.85
Epoch: 348  Step: 700  loss: 0.0218  accuracy: 0.85
Epoch: 349  Step: 0    loss: 0.0195  accuracy: 0.85
Epoch: 349  Step: 100  loss: 0.0211  accuracy: 0.85
Epoch: 349  Step: 200  loss: 0.0155  accuracy: 0.85
Epoch: 349  Step: 300  loss: 0.0157  accuracy: 0.85
Epoch: 349  Step: 400  loss: 0.0190  accuracy: 0.85
Epoch: 349  Step: 500  loss: 0.0247  accuracy: 0.85
Epoch: 349  Step: 600  loss: 0.0211  accuracy: 0.85
Epoch: 349  Step: 700  loss: 0.0188  accuracy: 0.85
Best accuracy: 0.864
```

Figure 14. Model A Log

```
Epoch: 348	Step: 0	loss: 0.0100	accuracy: 0.84
Epoch: 348	Step: 100	loss: 0.0141	accuracy: 0.84
Epoch: 348	Step: 200	loss: 0.0062	accuracy: 0.84
Epoch: 348	Step: 300	loss: 0.0197	accuracy: 0.84
Epoch: 348	Step: 400	loss: 0.0147	accuracy: 0.84
Epoch: 348	Step: 500	loss: 0.0104	accuracy: 0.84
Epoch: 348	Step: 600	loss: 0.0172	accuracy: 0.84
Epoch: 348	Step: 700	loss: 0.0134	accuracy: 0.84
Epoch: 349	Step: 0	loss: 0.0127	accuracy: 0.84
Epoch: 349	Step: 100	loss: 0.0172	accuracy: 0.84
Epoch: 349	Step: 200	loss: 0.0097	accuracy: 0.84
Epoch: 349	Step: 300	loss: 0.0113	accuracy: 0.84
Epoch: 349	Step: 400	loss: 0.0170	accuracy: 0.84
Epoch: 349	Step: 500	loss: 0.0082	accuracy: 0.84
Epoch: 349	Step: 600	loss: 0.0157	accuracy: 0.84
Epoch: 349	Step: 700	loss: 0.0135	accuracy: 0.84
Best accuracy: 0.844
```

Figure 15. Model B Log

```
Epoch: 348	Step: 0	loss: 0.0070	accuracy: 0.87
Epoch: 348	Step: 100	loss: 0.0062	accuracy: 0.87
Epoch: 348	Step: 200	loss: 0.0072	accuracy: 0.87
Epoch: 348	Step: 300	loss: 0.0099	accuracy: 0.87
Epoch: 348	Step: 400	loss: 0.0091	accuracy: 0.87
Epoch: 348	Step: 500	loss: 0.0065	accuracy: 0.87
Epoch: 348	Step: 600	loss: 0.0053	accuracy: 0.87
Epoch: 348	Step: 700	loss: 0.0111	accuracy: 0.87
Epoch: 349	Step: 0	loss: 0.0064	accuracy: 0.87
Epoch: 349	Step: 100	loss: 0.0057	accuracy: 0.87
Epoch: 349	Step: 200	loss: 0.0075	accuracy: 0.87
Epoch: 349	Step: 300	loss: 0.0076	accuracy: 0.87
Epoch: 349	Step: 400	loss: 0.0088	accuracy: 0.87
Epoch: 349	Step: 500	loss: 0.0098	accuracy: 0.87
Epoch: 349	Step: 600	loss: 0.0066	accuracy: 0.87
Epoch: 349	Step: 700	loss: 0.0082	accuracy: 0.87
Best accuracy: 0.877
```

Figure 16. Model C Log

```
Epoch: 348	Step: 0	loss: 0.0139	accuracy: 0.85
Epoch: 348	Step: 100	loss: 0.0102	accuracy: 0.85
Epoch: 348	Step: 200	loss: 0.0126	accuracy: 0.85
Epoch: 348	Step: 300	loss: 0.0165	accuracy: 0.85
Epoch: 348	Step: 400	loss: 0.0115	accuracy: 0.85
Epoch: 348	Step: 500	loss: 0.0152	accuracy: 0.85
Epoch: 348	Step: 600	loss: 0.0117	accuracy: 0.85
Epoch: 348	Step: 700	loss: 0.0195	accuracy: 0.85
Epoch: 349	Step: 0	loss: 0.0138	accuracy: 0.85
Epoch: 349	Step: 100	loss: 0.0105	accuracy: 0.85
Epoch: 349	Step: 200	loss: 0.0197	accuracy: 0.85
Epoch: 349	Step: 300	loss: 0.0127	accuracy: 0.85
Epoch: 349	Step: 400	loss: 0.0172	accuracy: 0.85
Epoch: 349	Step: 500	loss: 0.0251	accuracy: 0.85
Epoch: 349	Step: 600	loss: 0.0120	accuracy: 0.85
Epoch: 349	Step: 700	loss: 0.0209	accuracy: 0.85
Best accuracy: 0.864
```

Figure 17. Model Strided-CNN-C Log

```
Epoch: 348	Step: 0	loss: 0.0061	accuracy: 0.88
Epoch: 348	Step: 100	loss: 0.0065	accuracy: 0.88
Epoch: 348	Step: 200	loss: 0.0038	accuracy: 0.88
Epoch: 348	Step: 300	loss: 0.0062	accuracy: 0.88
Epoch: 348	Step: 400	loss: 0.0084	accuracy: 0.88
Epoch: 348	Step: 500	loss: 0.0071	accuracy: 0.88
Epoch: 348	Step: 600	loss: 0.0064	accuracy: 0.88
Epoch: 348	Step: 700	loss: 0.0052	accuracy: 0.88
Epoch: 349	Step: 0	loss: 0.0066	accuracy: 0.88
Epoch: 349	Step: 100	loss: 0.0027	accuracy: 0.88
Epoch: 349	Step: 200	loss: 0.0084	accuracy: 0.88
Epoch: 349	Step: 300	loss: 0.0030	accuracy: 0.88
Epoch: 349	Step: 400	loss: 0.0044	accuracy: 0.88
Epoch: 349	Step: 500	loss: 0.0068	accuracy: 0.88
Epoch: 349	Step: 600	loss: 0.0087	accuracy: 0.88
Epoch: 349	Step: 700	loss: 0.0056	accuracy: 0.88
Best accuracy: 0.891
```

Figure 18. Model ConvPool-CNN-C Log

```
Epoch 349, Iteration 0, loss = 0.0094
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 100, loss = 0.0160
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 200, loss = 0.0160
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 300, loss = 0.0094
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 400, loss = 0.0153
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 500, loss = 0.0088
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 600, loss = 0.0106
Checking accuracy on validation set
Got 850 / 1000 correct (85.00)

Epoch 349, Iteration 700, loss = 0.0125
Checking accuracy on validation set
Got 849 / 1000 correct (84.90)

Best accuracy: 0.869
```

Figure 19. Model ALL-CNN-C Log

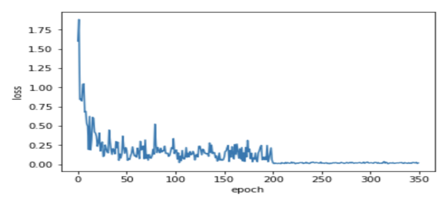

Figure 20. Model A Loss

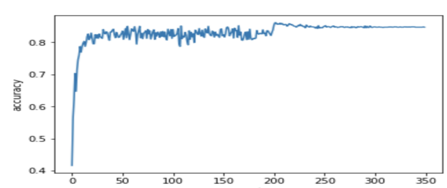

Figure 21. Model A Accuracy

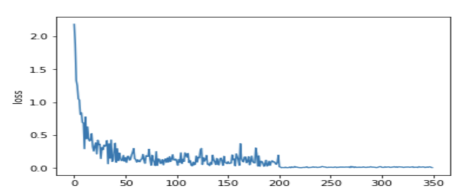

Figure 22. Model B Loss

6.3. Appendix 3

In classification task, the loss and accuracy figure of the six model is show as followed. In addition, we capture the log file.

References

[1] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

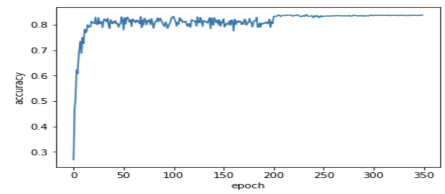

Figure 23. Model B Accuracy

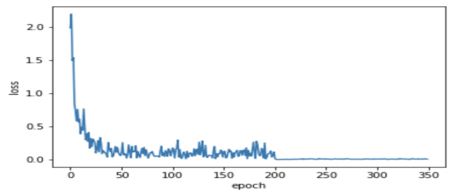

Figure 24. Model C Loss

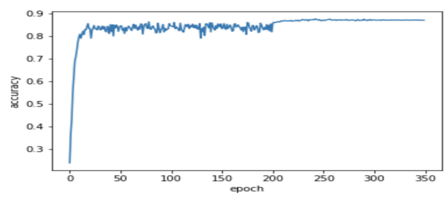

Figure 25. Model C Accuracy

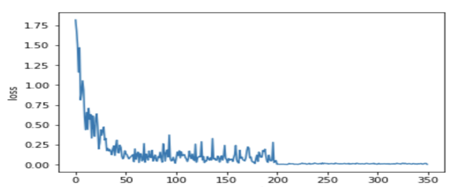

Figure 26. Model Strided-CNN-C Loss

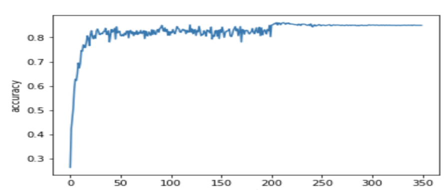

Figure 27. Model Strided-CNN-C Accuracy

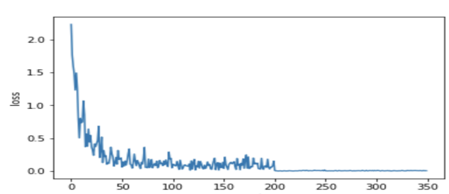

Figure 28. Model ConvPool-CNN-C Loss

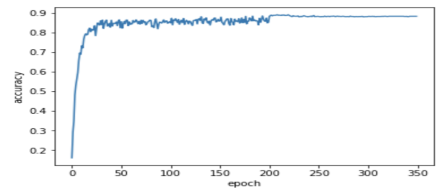

Figure 29. Model ConvPool-CNN-C Accuracy

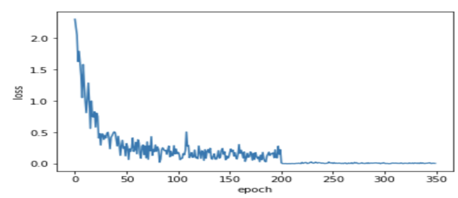

Figure 30. Model All-CNN-C Accuracy

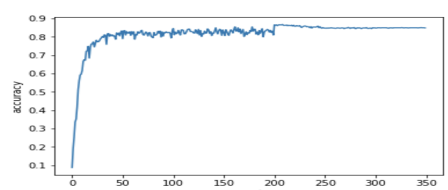

Figure 31. Model All-CNN-C Accuracy