Deep Learning Homework 2

Zheyu Zhang, Tsinghua University

A naive but complicated homework for DL.

1 Introduction

As we have learned CNNs in class, we're required to solve the problem of land cover image classification by modern convolutional neural networks (CNNs) [1].

EuroSAT is the dataset contains 10 categories of remote sensing images with different land covers including AnnualCrop, Highway, PermanentCrop, SeaLake, Forest, Industrial, Residential, HerbaceousVegetation, Pasture and River. We first randomly split the whole dataset into two parts, in which the first 80% is used for training while the remaining 20% is for testing [1].

Figure 1 shows how the dataset looks like.

2 Overview

THIS homework contains 3 Tasks.

In Task A, it is Large-Scale Learning. The task is traditional, I directly use ResNet-50 to solve it.

- Report training and test curves.
- Visualize the features before the last fully-connected layer using t-SNE.
- Leverage a proper neural network visualization toolkit to visualize some conv features.

In Task B, it is Medium-Scale Learning. The task is traditional, I designed My-Naive-Net and beats ResNet-50.

- Report training and test curves.
- Babysit it.

In Task C, I choose Task 5, that is Weakly-Supervised Learning. The task contains about 20% wrong labels. I use pre-train to recognize the wrong labels. Then the problem becomes simple.

- Report training and test curves.
- Plot the confusion matrix.
- Extra techniques.

3 Task A

Figure 2 shows the curves of the performance of ResNet-50 on Task 1.

Let's check if the features before Full-Connected Layer is good or not. Figure 3 shows the features embedded by t-SNE. It shows that the features seems almost linearly separable.

Then, the issue is that, how do we make sure the convolutional layers learned useful features? I use a *visualization toolkit* [2] to find out what a convolutional layer has learned. Figure 4 shows what a partical filter of each layers focus on. It shows that the model can focus on some important features.

4 Task B

Firstly, I design a net which is similar to LeNet. That is, trivially C-P-C-P-L-L. This doesn't work well compared with ResNet-50. Figure 5 shows the performance and the architecture.

Since it fits slowly and has bad expressive ability. The main idear to improve it is that, make more highways, and make more layers.

So I designed a model with lots of residual layers. With repect to Jurgen Schmidhuber, I call it My-highway-net instead of My-residual-net. It works very well and extremely beats ResNet-50. Figure 6 shows the performance and the architecture.

Then, it is necessary to Babysit it. The simplest way is Data-Augmentation. For each picture, we can rotate it or transpose it, so that each picture can generate 8(containing itself) pictures. Figure 7 shows the ways to rotate or transpose.

Well, we've done the Data-Augmentation. Figure 8 shows the curves after Data-Augmentation. Seeing the curves, may be the Optimization-

Figure 1. Samples in EuroSAT.

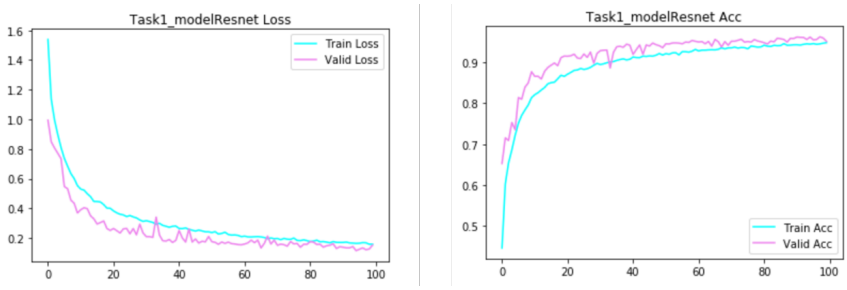

Figure 2. ResNet-50 on Task 1, the curves of loss and accuracy.

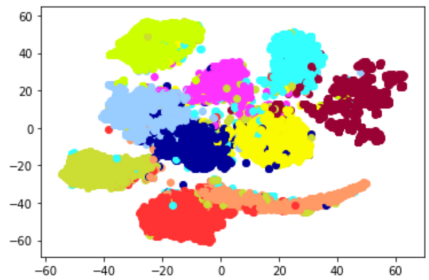

Figure 3. ResNet-50 on Task 1, features before FC Layer embedded by t-SNE.

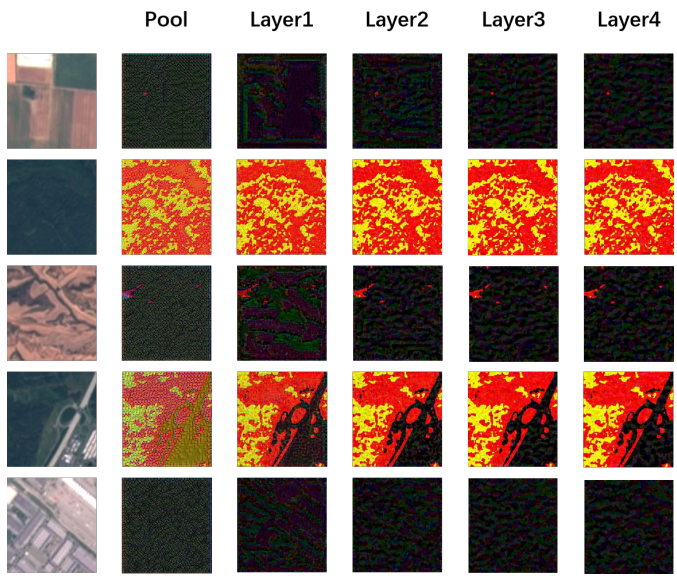

Figure 4. ResNet-50 on Task 1, what the fliters learned.

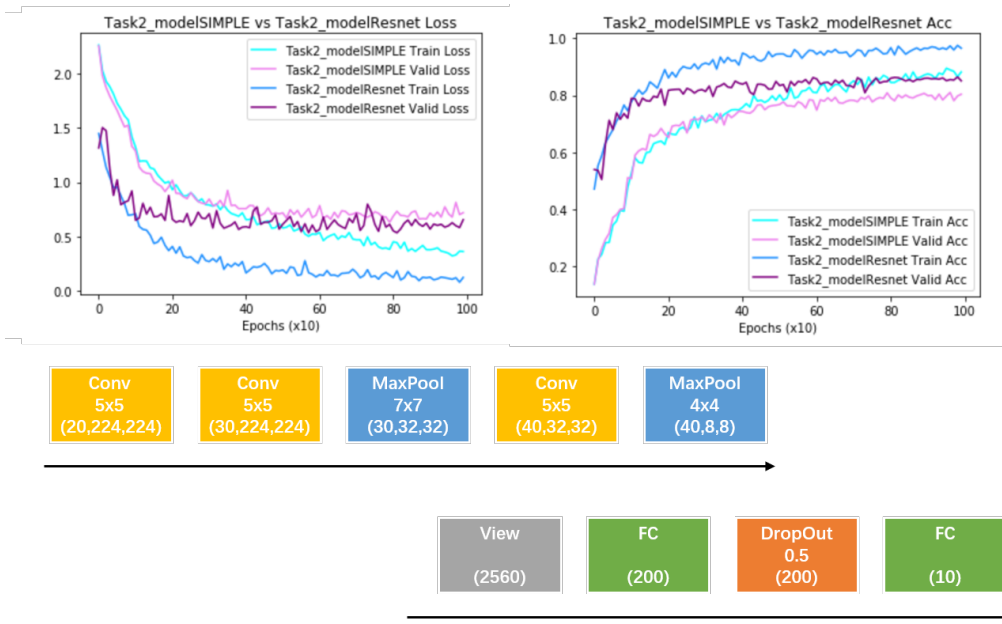

Figure 5. The simple net works badly on Task 2. (There are some Activation Layers omitted.)

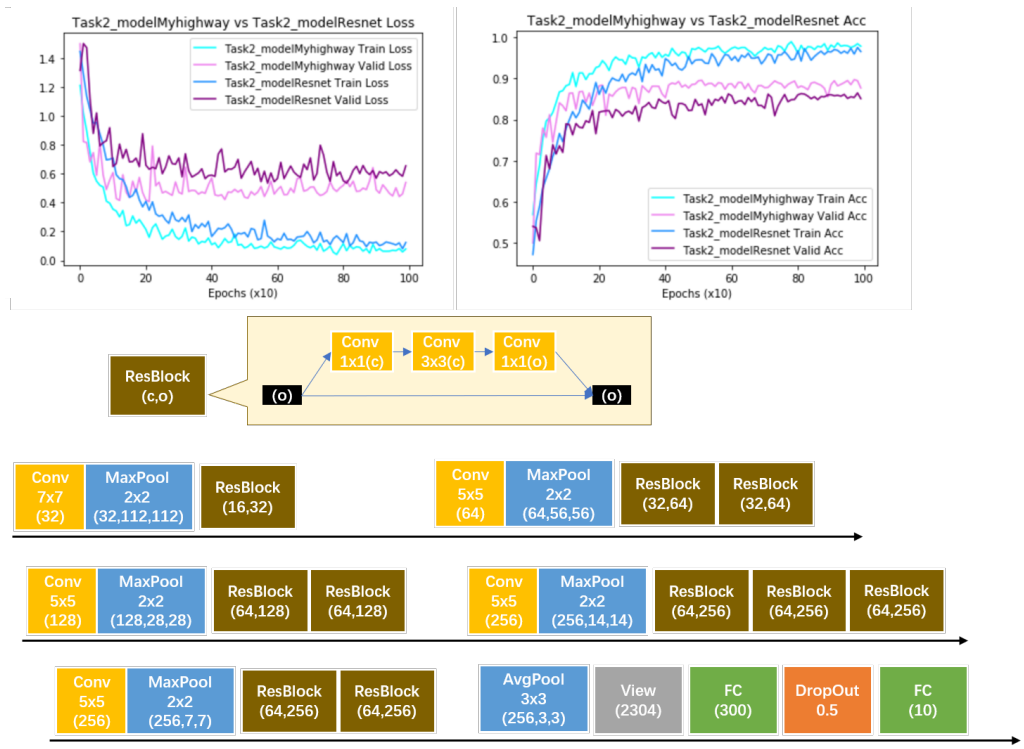

Figure 6. My-highway-net works well on Task 2. (Of course there are lots of Batch-Normalization and Activation Layers omitted.)

Algorithm could be improved since the curves shakes violently.

The next work I did is to babysit the Optimization-Algorithm. Due to my intuition, I change to Optimization-Algorithm to be Adam with Learning-Rate 0.0005. Finally, the model improved a litte. Figure 9 shows the improvement. During the babysit, I found out My-highway-net

Figure 7. Data-Augmentation method.

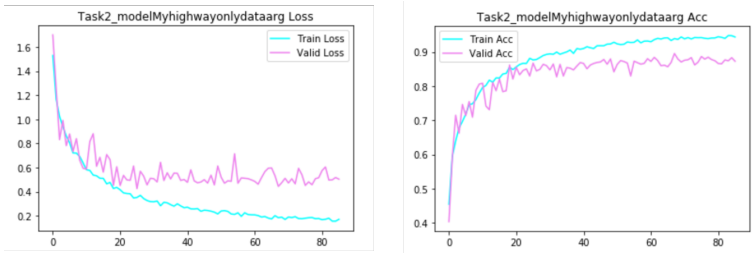

Figure 8. My-highway-net on Task 2 after Data-Augmentation.

is strong and has high robustness, so it is extremely hard to improve a lot on My-highway-net through babysit.

5 Task C

In this task, about 20% labels are wrong, but we can directly run model regarding all the labels right. Figure 10 shows the curves and confusion matrices. They are both optimized by SGD(0.001,0.9). It is easy to see that My-highway-net is more suitable for the task.

The next step is, how to make it better. We can directly use some traditional method such as Data-Augmentation, Ensemble-Learning. But a more efficient way is that, use pre-trained model to delete the wrong labels. (I mean, delete something looks completely wrong.)

Due to my intuition, it is efficient to use the model trained by the whole training data to verify the wrong labels. Though the whole training data contains 20% wrong labels, as long as the model has high generalization ability and low fitting ability, it can verify a great number of wrong labels. So I use My-highway-net trained before, to delete the wrong labels and get 'Good' dataset. Then, train another My-highway-net on the 'Good' dataset. Figure 11 shows the curves on 'Good' dataset. It shows that the method works.

Acknowledgements

Thanks to MatPool® for selling GPU to me.

References

[1] *Homework 2: Convolutional Neural Networks and Beyond*, Deep Learning (84100343-0), Spring 2021, Tsinghua University.

[2] *Convolutional Neural Network Visualizations*, github, https://github.com/utkuozbulak/pytorch-cnn-visualizations.

GPU is expensive. Note that, it is better to run a model longer until a model is completed fitted and stop decreasing loss. But as you know, GPU is quite expensive. A more effective way is to stop while it looks really hard for Valid-Accuracy to increase.

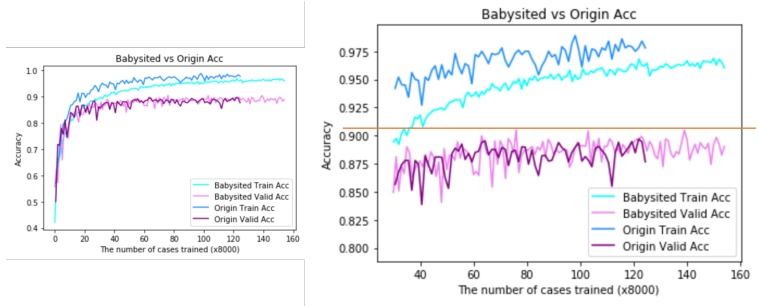

Figure 9. My-highway-net on Task 2 after babysit, improved a little.

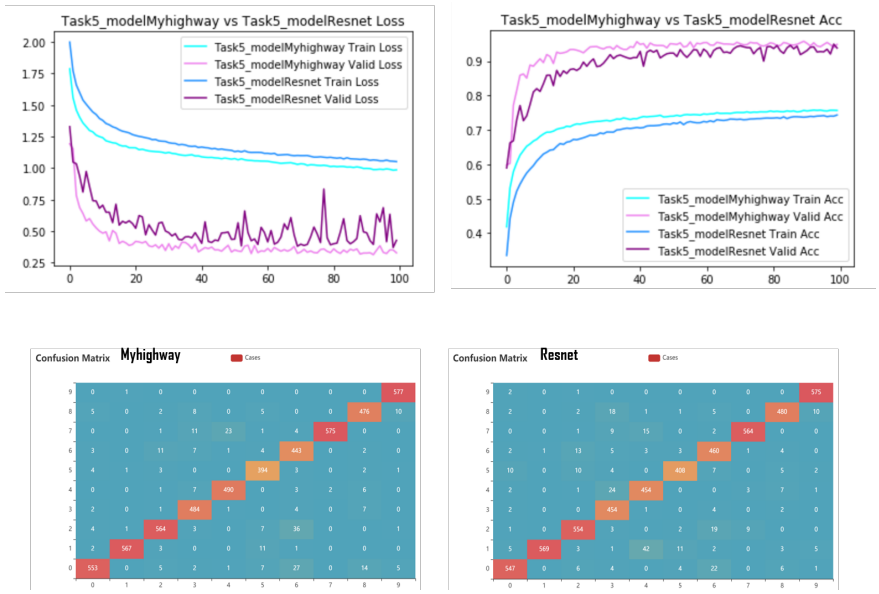

Figure 10. My-highway-net and ResNet-50 on Task 5, simply regarding all labels right.

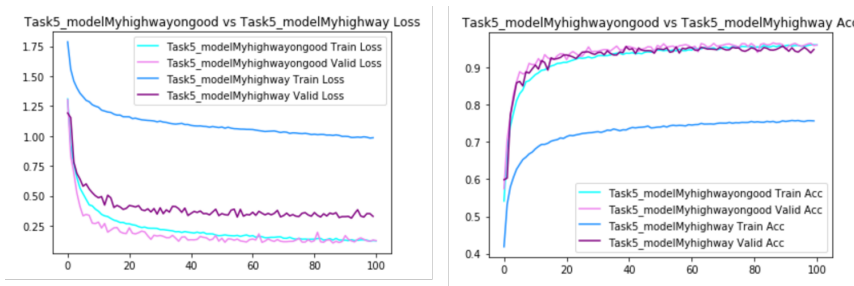

Figure 11. My-highway-net on Task 5, after deleting some wrong labels.