Image Classification Based on Deep Convolutional Network

Sai Wu
PID:A53212368
saw044@ucsd.edu

Xinyang Wang
PID:A53204245
xiw383@ucsd.edu

Wen Liang
PID: A53214852
wel245@eng.ucsd.edu

Hua Shao
PID: A53220045
h5shao@eng.ucsd.edu

Abstract

In this assignment, Convolutional Neural Network has been explored to classify CIFAR 10 dataset. The three model we implemented with 5 or 6 convolutional layers and 1 or 2 max pooling layers and 3 fully connected layers. Layers, Kernel size, depth and max pooling layers are experimented. The batch normalization, Xiaver weight initial, Adam optimizer are also tried in our implementation. The best test accuracy we achieve is 87.47%. Leaderboard teamname is HappyNewYear. In the second part, pre-trained CNN VGG16 is used as a pre-stage network.Also, as an experiment, we take an intermediate convolutional layer as an output and use it as input for a softmax layer to test its performance. FInally, to better understand what's going on in convolutional layer, we visualize the output from the first and the last convolutional layer. We also analyze the property of temperature softmax.

1 Introduction

In this assignment, we used PyTorch to implement convolutional neural networks to explore the model with different layers of convolution, pooling and fully connected layer, different techniques, optimizers in the sections2. PyTorch is a Python package can provide tensor computation (like NumPy) with strong GPU acceleration and deep neural networks built on a tape-based autograd system. Which is easy to use to build some deep neural network model. Also, using the GPU we can training the neural network with more parallel computation. Then, we can train and tune a model with higher efficiency.

In the sections 2, we take the advantage of pretrained CNN model using transfer learning. In practice, it is hard to train a CNN from the random initialization, because the size of dataset limits a lot. Equipped with pretrained filters in VGG16, the model can solve a problem with a small dataset and converge faster. Then, we visualized the filters and feature extraction process in the section 3.1 and 3.2.

2 Deep Convolutional Network for Image Classification

2.1 Dataset Preprocessing

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The training set is shuffled and

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

divided into 45000 images for training and 5000 images for validation.Data augmentation can effectively enlarge the size of dataset, improve the accuracy of model, avoid overfiting and imporve the robustness to different angle, size, occlusion and some other situations. Before training the classifier, we did corpping, scaling and horizontally flipping to enlarge the training dataset. In addition, data normalization is applied after data augmentation.

2.2 Experiment with Different Methods

The first experiment we did is comparing the effect of number of convolution layers in CNN mode. The result shows that with more layers and proper number of units in each layer can achieve higher accuracy. However, to avoid out of memory issue, we only increase our number of convolution layers to 6. Batch normalization, Xavier initialization and Adam Optimizer are also experimented during our process. Since Xavier works positively on activation function with flatten regions such as sigmoid function which has two flatten region. It does not have much improvement on ReLU activation function which has only one flatten region. Batch normalization accelerates the deep network convergence by normalizing the internal covariate shift and rearranging the mean and variance. These process regulates the distribution of the output of current layer.The experiment result shows that it works well on our CNN model. It can obviously increase the training and validation accuracy by 3% to 5%. It can almost increase 10% accuracy at the beginning of training so that it can also short the cost time of convergence. However, it does not have much improvement when implement in fully connected layer, so only implement it in convolutional layers. Moreover, Adam optimizer is an extension to stochastic gradient descent, it can improve the efficiency during our training.

2.3 Summary of 3 Models

Figure 1, figure 2 and figure 3 demonstrated the structures of three models we implemented. Since the pixel for each image is tiny, so choose smaller kernel size to implement our model. The kernel size in our model is 3×3. The stride of our CNN layer is commonly 1. Padding is 1.

Model 1 contains five convolutional layers and 3 fully connected layers and one max pooling layers. After every convolutional layer, ReLU is applied. At the same time, batch normalization is applied in convolutional layer as well. In the third convolutional layer, we used stride of 2 instead of 1. Because this model only have one max pooling layer after five convolutional layer, so use stride of 2 to control the numbers of outputs. The depth of first convolutional layer is 46. The following convolutional layer is incremented by 2 times which are: 92, 184, 368, 736. It takes 45 epoch to convergence. The test accuracy of the Model 1 is 83.34%. The training accuracy is 87.87%.

Model 2 contains 6 convolutional layers, 3 fully connected layers and two max pooling layers. Same as Model 1, after every convolutional layer, ReLU is applied. At the same time, batch normalization is applied in convolutional layer as well. Every three convolutional layers, one max pooling is followed. The depth of first convolutional layer is 46. The following convolutional layer is incremented by 2 times which are: 92, 184, 368, 736, 1472. It takes 40 epoch to convergence. The test accuracy is 87.47%. The training accuracy is 96.89%.

Model 3 contains 6 convolutional layers, 3 fully connected layers and one max pooling layer. Same as before, after every convolutional layer, ReLU and batch normalization is applied. Different with Model 2, this model only has one max pooling layer applied after 6 convolutional layers. The depth of each convolutional layers is: 46, 92, 184, 368, 736, 1472. It takes 40 epoch to convergence. The test accuracy is 86.72% and training accuracy is 96.45%.

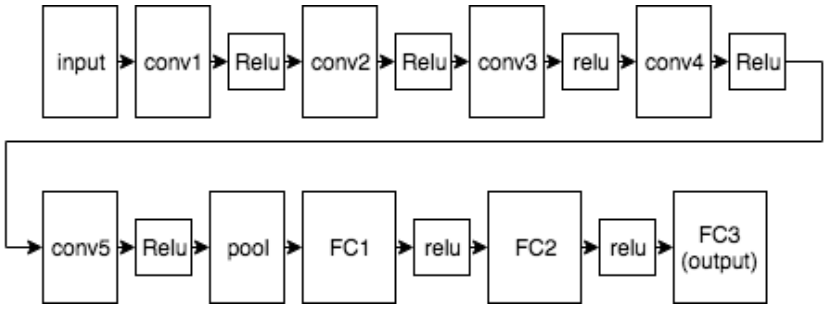


Figure 1: Structure of CNN (Model 1)

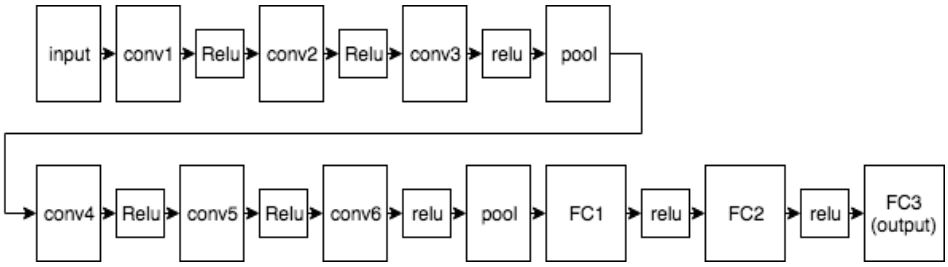


Figure 2: Structure of CNN (Model 2)

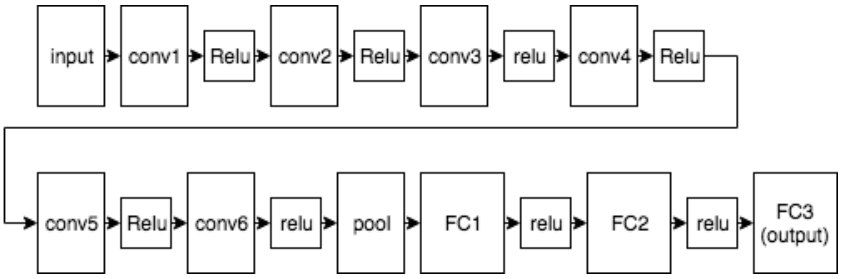


Figure 3: Structure of CNN (Model 3)

2.4 Plot Results of 3 Models

Figure4,5 and 6 shows the train, validation and test results of our 3 models. The test accuracy of the Model 1 is 83.34%. The training accuracy is 87.87%. For model 2, the test accuracy is 87.47%. The training accuracy is 96.89%. For model 3, the test accuracy is 86.72% and training accuracy is 96.45%.

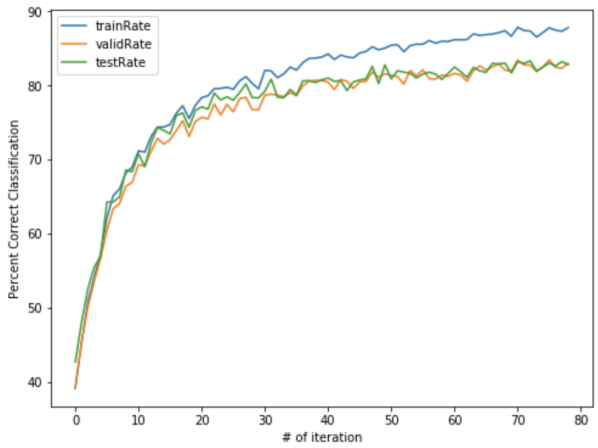


Figure 4: Training, Validation and Test Accuracy (Model 1)

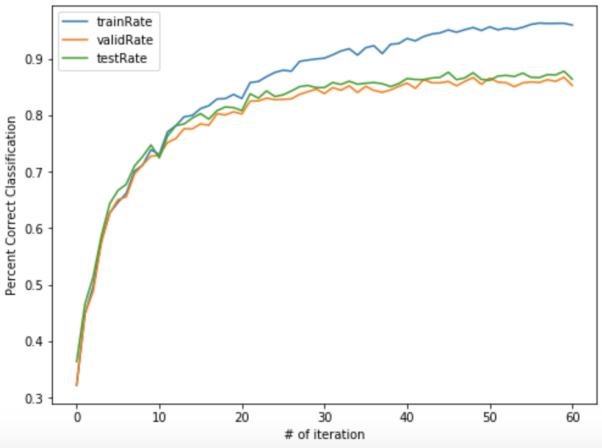


Figure 5: Training, Validation and Test Accuracy (Model 2)

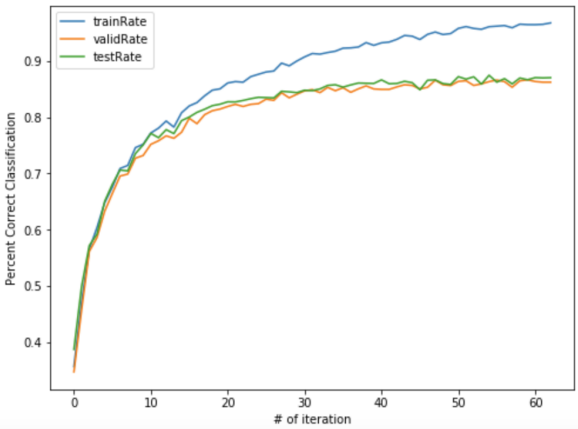


Figure 6: Training, Validation and Test Accuracy (Model 3)

2.5 Discussion

During our training with different model, it is clearly that a model with more convolutional layers can have higher accuracy on validation and test set. In our four convolutional and 2 fully connected model, it can achieve round 80% accuracy. When we tried 5 or 6 convolutional layers with 3 fully connected layers, the accuracy is over 85%. However, the convergence time with more layers is slower where it took almost 2 hours for model 2 and model 3 to convergence.

As I states in the experiment methods part, the batch normalization works well in our model which improves almost 2% or 3% accuracy. It also improve the cost time of convergence. The Adam (Adaptive Momentum Estimation) is a kind of modification of RMSprop with momentum. For parameter update, it combines the momentum method and adaptive learning rate method. The momentum averages the gradients in each step, the root mean square part adapt the moving step size. Also, the bias correction mechanism can make the convergence faster in first several iterations. During the training process, Adam works positively and improves the efficiency. However, since Xavier initialization is applied on linear models, it does not show improvement on our model. At the same time, data augmentation can improve around 2% test accuracy since it make the training set more variant.

The batch normalization improved the accuracy on test set by 2%. This technique makes our model much more robust. One reason is that, for deep neural network, there exists much more randomness in the output of each layer and the distribution is more uncontrollable. Also, the "covariate shift" (which means that when you learned a X to Y mapping, if you change the distribution of X, then you have to retrain your learning algorithm) forces the later layers to adapt to the distribution of its previous layers. Therefore, using the batch normalization, we can regulate the distribution of output then reduce the covariate shift and let the model put more concentration on extracting features and explaining variance.

We also tried to use different kernel sizes for filters. The model 1 with 5×5 kernel has 80.17% accuracy on test set, which is 3% lower than model 1 with 3×3 kernel. The reason is that the dimension of images in CIFAR dataset is rather small. In addition, using unreasonable big kernel size can make the model harder to converge and cause more computation burden. However, because every dataset has different properties, different colors and different data sources, a good way to decide the kernel size is doing experiments and comparing different choices. Moreover, we also found a technique that we can mix the output from filters with different kernel sizes which can provide diverse features.

3 Transfer Learning

3.1 Transfer Learning with VGG16

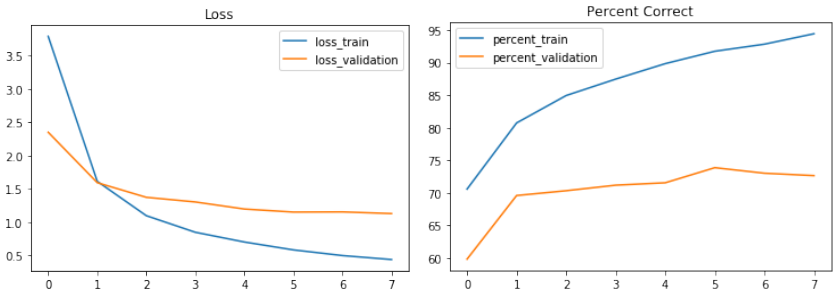
3.1.1 Repeat Training on dataset

First, we will introduce our training process of transfer learning. After downloading vgg16 classifier, we replace the softmax layer to a new softmax layer which is used for tuning our dataset. We initialize our input data by normalizing data into normal distribution, and scale it into 224*224. We use Adam optimizer for back propagation. Parameters are shown below in our model:

- learning rate of Adam optimizer: 0.0001
- Training epoch is 8
- Convolution part of the net work is pretrained model of vgg16.
- 32 samples per category.

Training for 8 epochs and we get the plot.

After eight epoch of training, the train loss goes down to 0.441 and the accuracy goes up to 74.85%. The vgg16 network is trained by imageNet which has 1000 classes. and we retain the parameter in the pretrained network. Its performance is acceptable for a network not trained on the Caltech256 dataset. With the increase of the iteration, the loss goes down.



(a) Loss vs epochs (b) Accuracy vs epochs

It is clear to see that the loss will always decrease, and the accuracy will always increase with the increase of epoch until it encounters overfitting. As for our inference, it is always better to have more training samples, though when the number of sample increases to a certain level, the effect of adding more samples can be very small. However, for epoch, the network might be overfitted if the network is trained too many times.

3.1.2 Visualize Filter and Weights

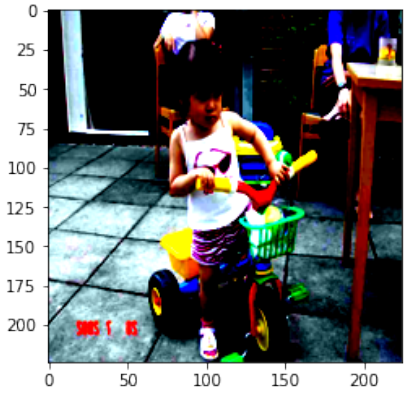


Figure 8: original image

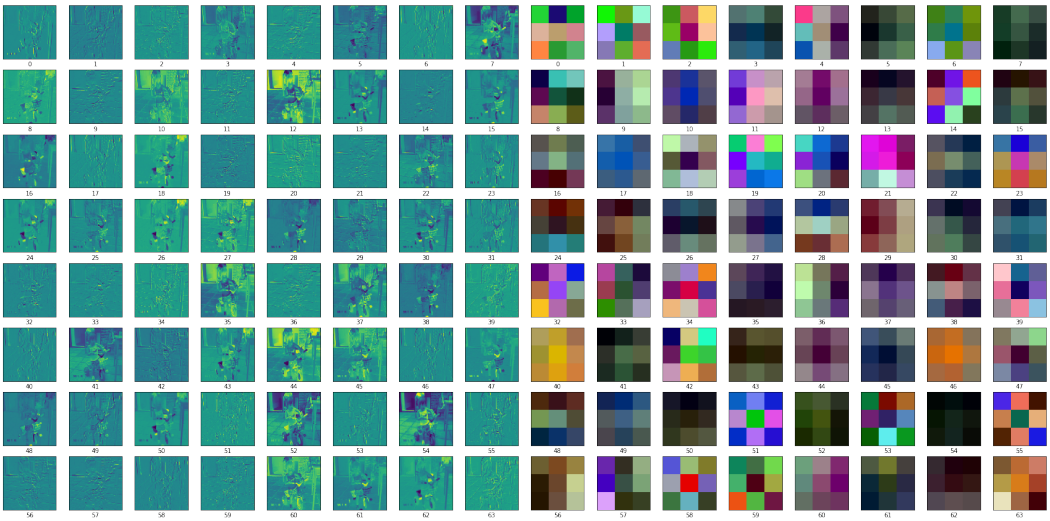
Original image is shown in Figure 8, the results after one-layer convolution are shown in Figure 9a. We can see that, images in Figure 9b are the results through some simple filters. Filters of the first layer are shown in Figure 9b.

We can see that in the no.1 and no.2 subfigures in Figure 9a, the feature map shows that this filter detect the edge. Filter no.1 no.2 can detect the edge of the original input from different directions. Sub-figure no.54 retains most the characteristics of inputs, we can see from the filter that every weights of the filter are similar values.

However, the output (Figure 10) from the last convolutional layer is totally different. Almost many of them pass nothing to the next layer, and this is where the most distinctive features are ”filtered” out, and make the picture be classified.

3.2 Feature Extraction

In order to avoid the memory overflow, we only use one fully connected layer before softmax layer and mini-batch size is 5. Figure 11 is the plot of loss and accuracy for 3 convolutional blocks with fully connected layer and softmax. The Accuracy reaches about 28% after 30 epochs. Our learning rate is 0.000001. Figure 12 is the plot of loss and accuracy for 4 convolutional blocks with fully connected layer and softmax. The Accuracy reaches about 49% after 30 epochs. Our learning rate is 0.000001.



(a) One Layer Results Feature Map (b) Filters

Figure 9

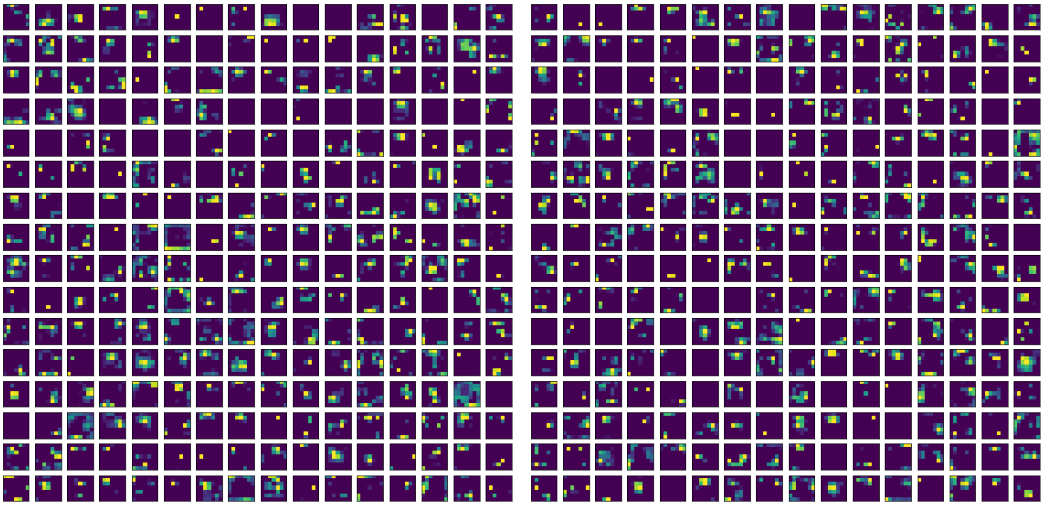


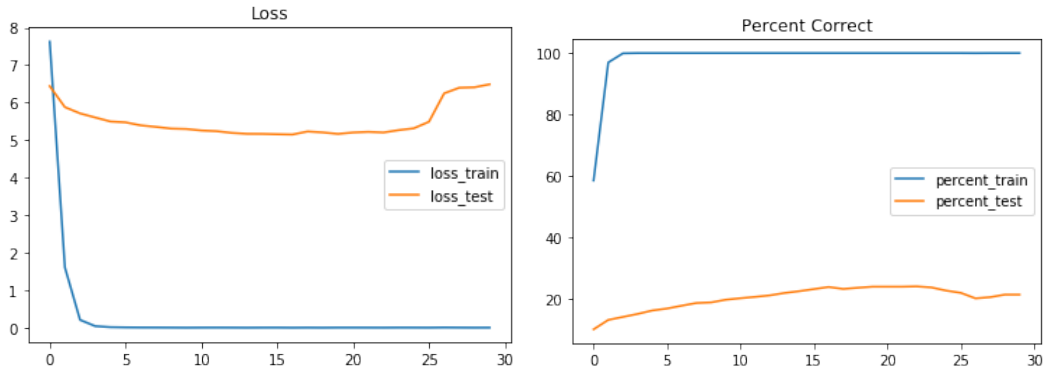
Figure 10: Feature maps from the last layer

Therefore, with these being said, we know that the performance of feature extraction using intermediate convolutional layer relies on the number of training samples and the depth of the neural network. We tried using more training samples, in which we observe the decreasing of loss and increasing of accuracy, but it would take too long time to train.

4 Conclusion

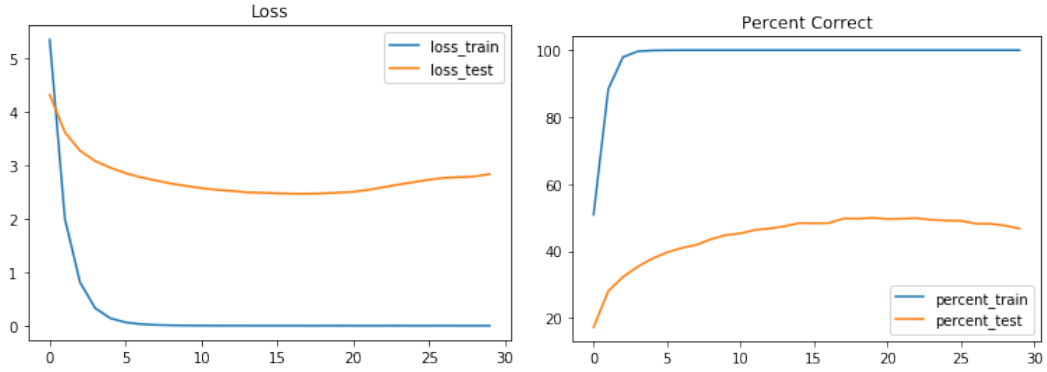
4.1 Part 1

In this section, we implemented 3 different structure of CNN, which has different number of convolution layers, pooling layers and fully connected layer. You can find the detailed settings in section 2.3 and the accuracies on testset for each model are 83.34%, 87.47% and 86.72%. The results shows that more convolution layers can let model get higher level features, get more ability to solve the classification problem and get better performance. And pooling layer can reduce the input dimen-



(a) Loss vs Epochs (b) Accuracy vs Epochs

Figure 11: 3 convolution blocks



(a) Loss vs Epochs (b) Accuracy vs Epochs

Figure 12: 4 convolution blocks

sions. Moreover, we tried Xavier initialization, batch normalization, Adam optimizer and different size of kernels. The result shows that the batch normalization can improve the accuracy a lot, the Adam optimizer can speedup the training process and we need experiments to decide a proper kernel size.

4.2 Part 2

In this assignment, we learned to use library and frame to build our nerual network, which can increase the speed of training. The most important thing of the performance of a neural network is the depth and the structure of the neural network. Good combination of convolutional layer, pooling layer and Linear layer can result in a good performance of the network. To increase the accuracy in a certain neural network, generally we have two ways to do. The first one is to increase training epoch, and the second one is to increase training samples. However, the accuracy can never reach a very high level. As for increasing training epoch, it can easily be seen in the figure that, the accuracy will eventually be flattened. Also for increasing training samples, with the increment of samples for each category, the accuracy increases logarithmically. Therefore, as a trade-off, we always need to balance between accuracy and training time, since the more epoch we train or the more training samples we use, the more time we will need.

5 Contribution

Wen Liang and Hua Shao mainly work on the section 1. They worked on experiments on CNN model with different convolution, pooling and fully connected layers and explored effects of different techniques and optimizers.

Sai Wu and Xinyang Wang mainly work on the Part 2. We work on the Transferring Training and the Feature selection of the VGG16. and help coding the part 1.

We all completed this report together.