Faster RCNN for cifar 10 Experiment Report

Haoyuan Zhang (Robotics, GRASP Lab, UPenn)

November 19, 2017

Abstract

The experiment is aimed at implementing and analyzing several well developed networks for feature extraction, localization and recognition. We are supposed to experiment and analyze three base networks including ConvNet, MobileNet and ResNet. After developing the base networks, we made use of features to localize the object in an image. Finally, we completed to implement a simplified version of Faster RCNN.

1 Base Networks: ConvNet, MobileNet, and ResNet

Features play important roles in computer vision research to represent images, like SIFT and HOG features. Nowadays, researchers turn to construct deep network to extract features. In this part, we try three different base networks and compare their performance and efficiency using the CIFAR-10 dataset. In addition, I use PyTorch and GPU for this part.

1.1 ConvNet Based Structure

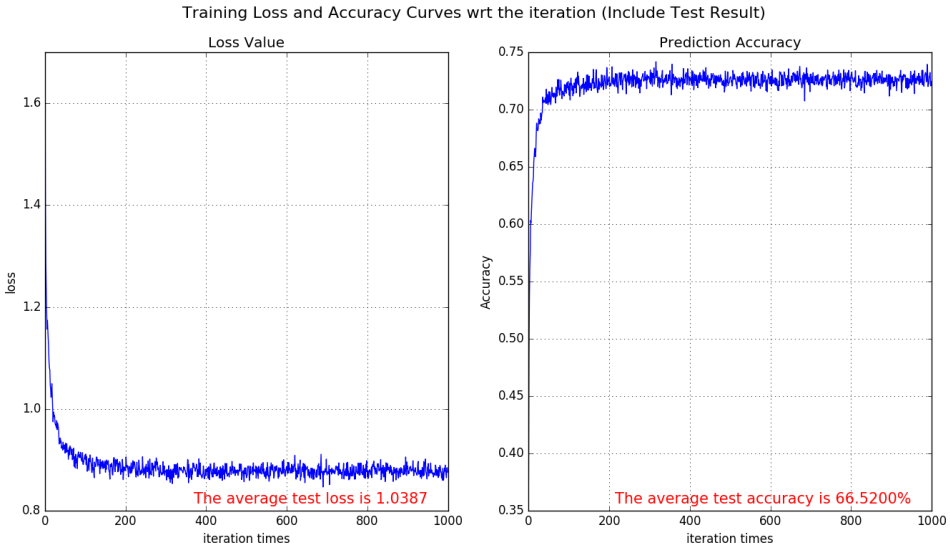

Figure 1: Training loss and accuracy curves (ConvNet)

The above figure shows the training process of ConvNet (including training loss and accuracy). For the experiments, the total **500** epoch iterations use **2418.692** s.

For the multiplications computation in the standard convolution layers, we have the below general equations. (We assume the padding strategy is **SAME**)

$$Multi_{Times} = K_h \times K_w \times im_h \times im_w \times C_{in} \times C_{out} \tag{1}$$

Here, K_h and K_w represent the height and width of filter respectively, similarly, im_h and im_w are the size of input data. C_{in} and C_{out} show the input and output channel numbers. According to the setting of ConvNet structure, the total number of multiplications in convolution layers is

$$5^2 \times 32^2 \times 3 \times 32 + 5^2 \times 16^2 \times 32 \times 64 + 5^2 \times 8^2 \times 64 \times 128 + 5^2 \times 4^2 \times 128 \times 256 + 3^2 \times 2^2 \times 256 \times 512 = 46497792 \quad (2)$$

1.2 MobileNet Based Structure

Here we use the MobileNet block to replace the first four ConvNet blocks, so that obtained the below training processing figure.

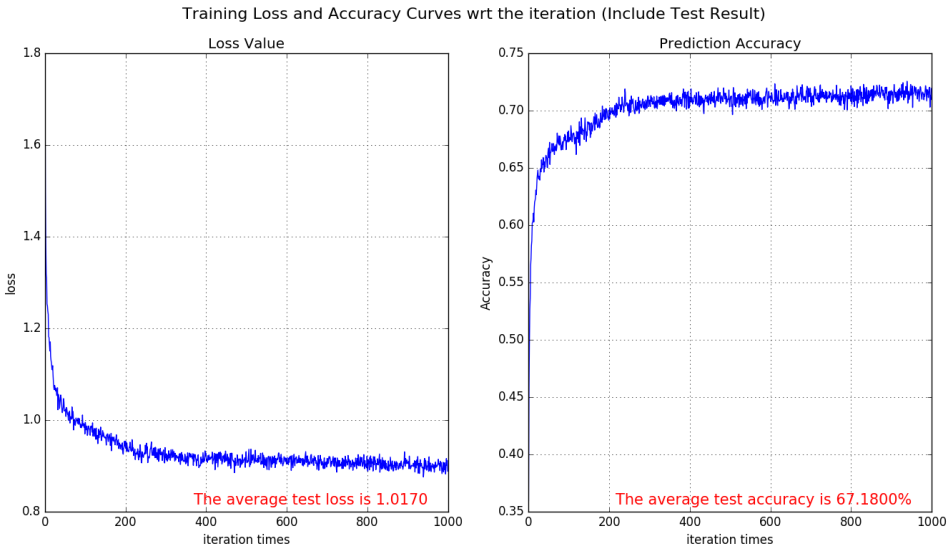

Figure 2: Training loss and accuracy curves (MobileNet)

Same as ConvNet training, the total max iteration times is **500**, it costs total **4186.518** s.

For the multiplications computation in the MobileNet convolution layers, we have the below equations. (We assume the padding strategy is **SAME**)

$$Multi_{Times} = K_h \times K_w \times im_h \times im_w \times C_{in} + 1 \times 1 \times im_h \times im_w \times C_{in} \times C_{out} = im_h \times im_w \times C_{in} \times (K_h \times K_w + C_{out}) \quad (3)$$

So finally the total multiplication times in those convolution layers is **6824960**. Therefore, the multiplication times of MobileNet is extremely less than ConvNet's. However, according to the experiment time results, the training time of MobileNet based structure is longer than ConvNets, the reason is, within each MobileNet Block, it introduces more batch normalization and relu operations, therefore, the total time cost is larger than previous one.

1.3 ResNet Based Structure

Now we use the ResNet block to replace the first four original ConvNets. Below shows the training process and final test accuracy.

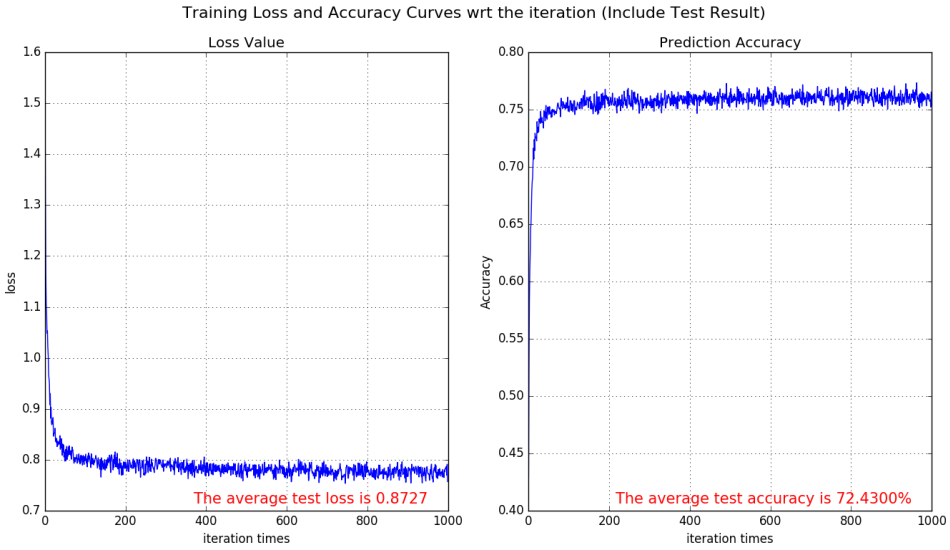

Figure 3: Training loss and accuracy curves (ResNet)

The total **500** iterations spent **4739.502** s.

For the multiplications computation in the ResNet convolution layers, we have the below equations. (We assume the padding strategy is **SAME**)

$$Multi_{Times} = im_h \times im_w \times C_{out} \times (10 \times C_{in} + 9 \times C_{out}) \tag{4}$$

So finally the total multiplication times in those convolution layers is **59179008**. Therefore, the multiplication times of ResNet the most one with the longest training time cost as well.

1.4 Compare and Explanation

According to the training process figures for all three base nets, the loss and accuracy will fluctuate a lot, which means is not easy to converge. ResNet will give me the highest test accuracy and lowest test loss. The test accuracy of ConvNet and MobileNet performs similarly. In a word, in terms of training and performance efficiency, the MobileNet might be the best one among those three base structures.

2 Region Proposal Networks

In this section, we build a region proposal network step by step with the base net in the previous section. RPN is a part of Faster RCNN for localizing objects in an image. We implemented a simplified version of region proposal network which consists of only one set of anchors. Here, we take the CIFAR-100 data as background and CIFAR-10 data as objects for detection and classification.

2.1 Proposal Classifier Branch

At the top of base net, we introduce two more convolutional layers for the proposal classifier task, that is, a binary classification task to figure out whether the current position contains an object or not.

Here we use a ground truth mask with 6x6 size for loss computation, I set a higher threshold (0.75) to filter out positive objects and lower threshold (0.25) to get negative training objects. Ideally, after training, the position that contains an object should have the feature magnitude close to 1 and non-object places' value almost equal to 0. In addition, we don't take the backgrounds into consideration. Below shows the training process curve as well as final test accuracy.

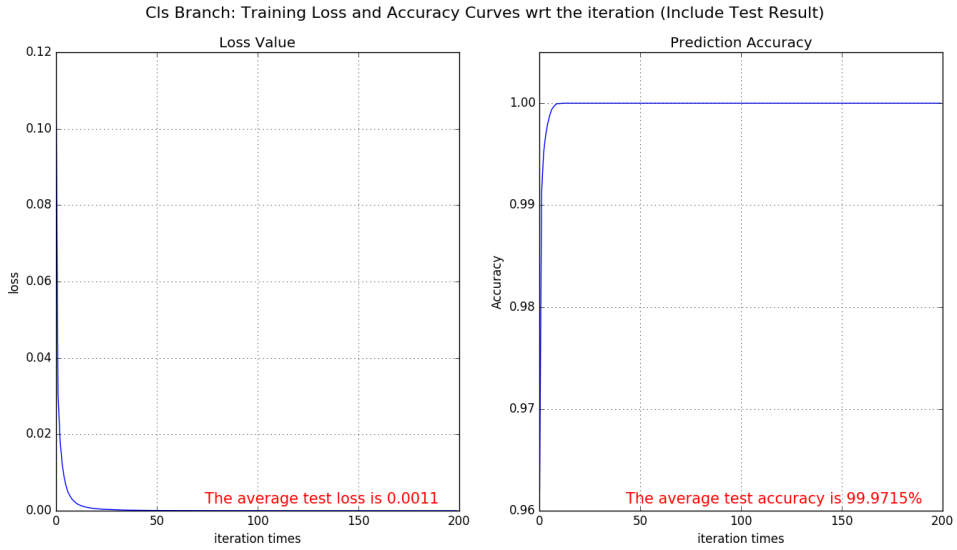

Figure 4: Training loss and accuracy curves (Proposal Classifier Branch)

According to the result curve, the training for proposal classifier converges pretty fast (around 6 or 7 epochs) and the final test accuracy is also reasonable.

2.2 Proposal Regression Branch

On the top of intermediate layer, we attach a new branch for proposal regression, which is parallel to the proposal classifier. The proposal regression is to determine the location information of objects. Given the ground truth and feature map from intermediate side, we compute the smooth L1 loss for proposal regression branch training. In addition, we still keep the proposal classifier one so that construct a complete simplified RPN. Below shows the training process and test accuracy for the proposal regression.

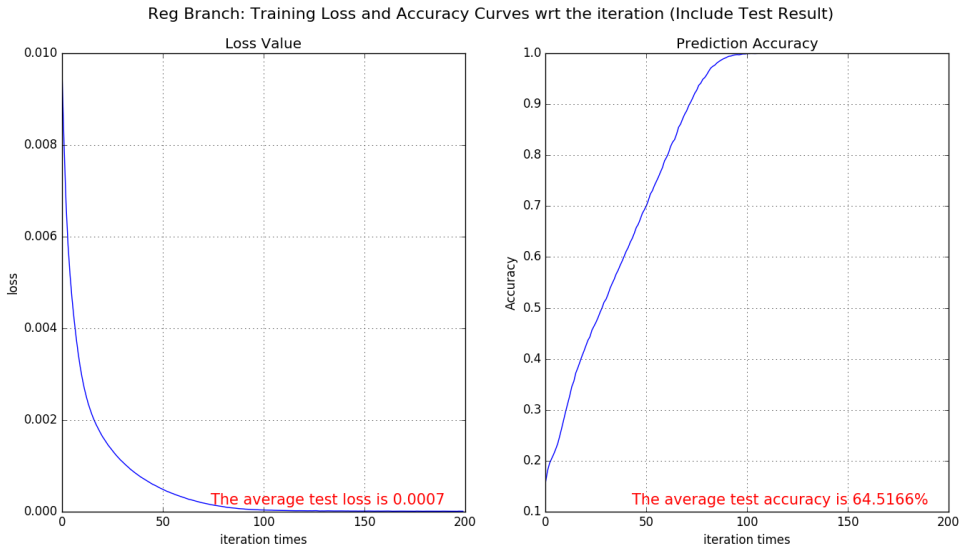

Figure 5: Training loss and accuracy curves (Proposal Regression Branch)

3 Faster RCNN

With the base structure of RPN, we continue constructing the Faster RCNN structure. Specifically, we apply the spatial transformation layer to transform feature map from Conv4 to locate the object position, then attach the fully connected layers and softmax to obtain the loss and class recognition.

3.1 Spatial Transformer

The spatial transformer layer will transform the features of proposal into the final object classifier. Good location estimation can help the recognition accuracy. Below shows four transformer results from the STN, I also compare it with the results generated from opencv built-in function.

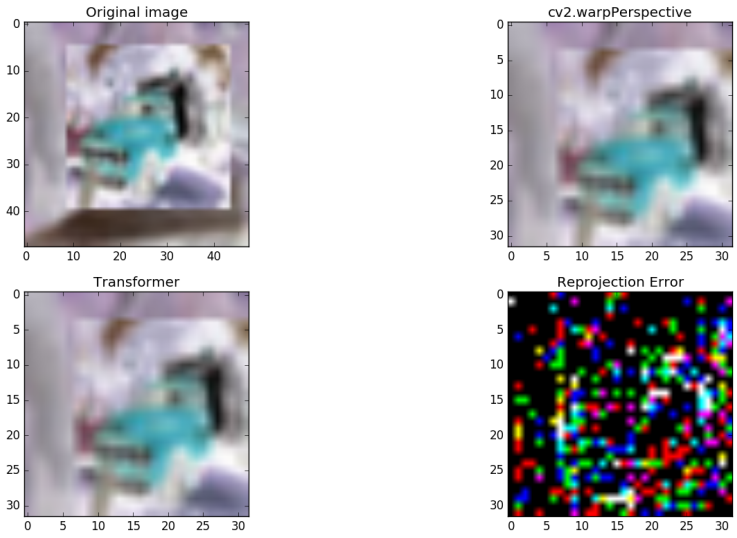

Figure 6: Spatial Transformer Results 1

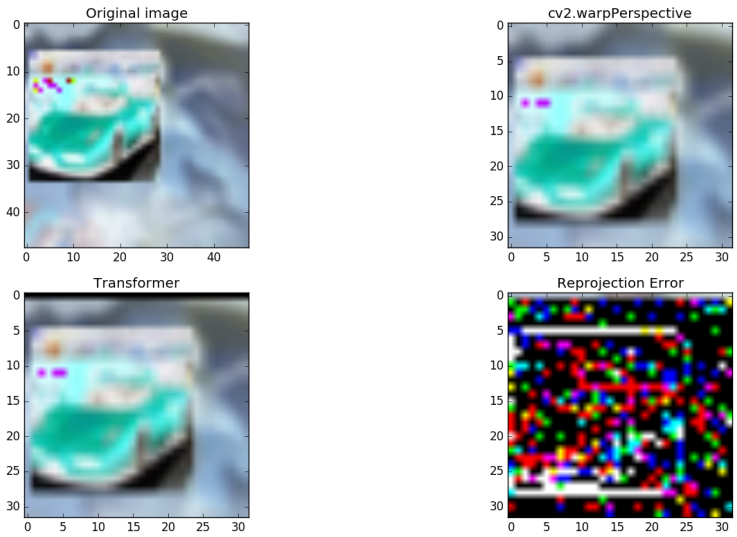

Figure 7: Spatial Transformer Results 2

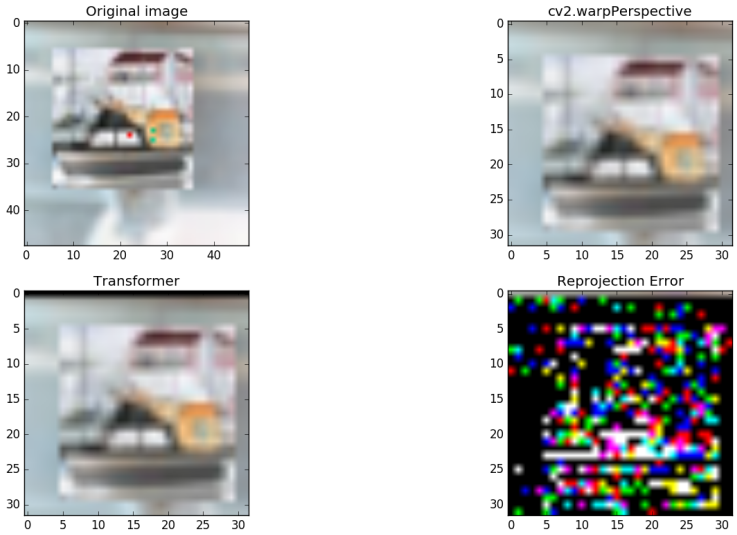

Figure 8: Spatial Transformer Results 3

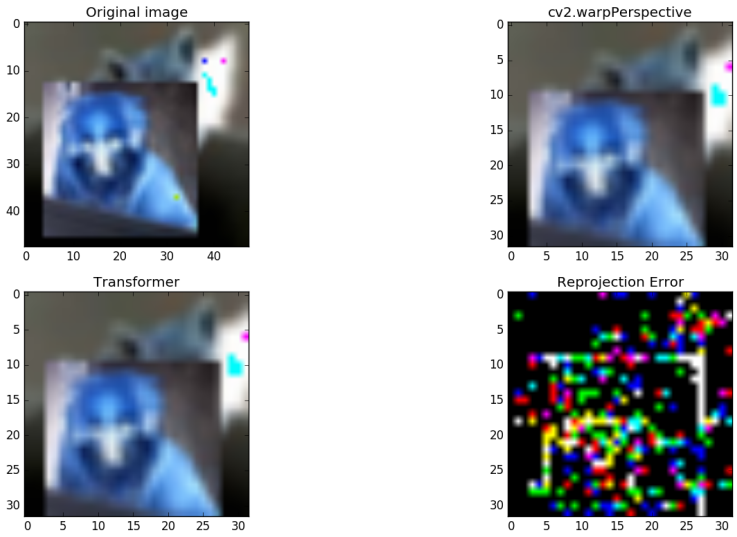

Figure 9: Spatial Transformer Results 4

According to four figures above, it's quite obvious that the spatial transformer can locate the object position more accurate which do a lot of favor to the classification.

3.2 Faster RCNN with ConvNet Base

Here, we use the feature map from proposal classifier branch, extract the position with the max value as object location, then use the feature map from proposal regression branch to obtain the bounding box information such as center coordinates and width, we use those parameters to generate a affine transformer matrix to map the feature map. Then attach two fully connected layers to get classification loss to train the network. Below figures shows the loss and accuracy of obj, proposal cls and reg branches during the training process, as well as including the test accuracy.

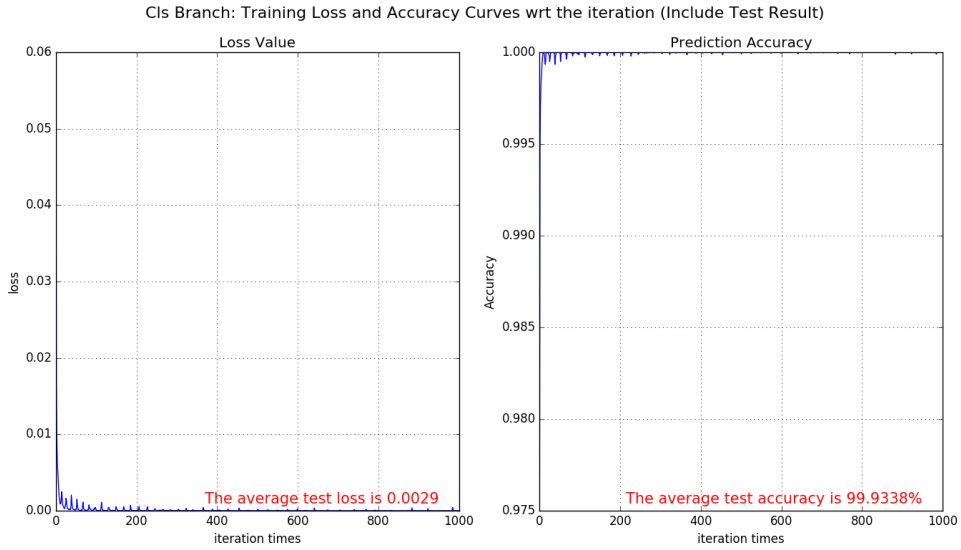

Figure 10: Training loss and accuracy curves (Proposal Classifier Branch)

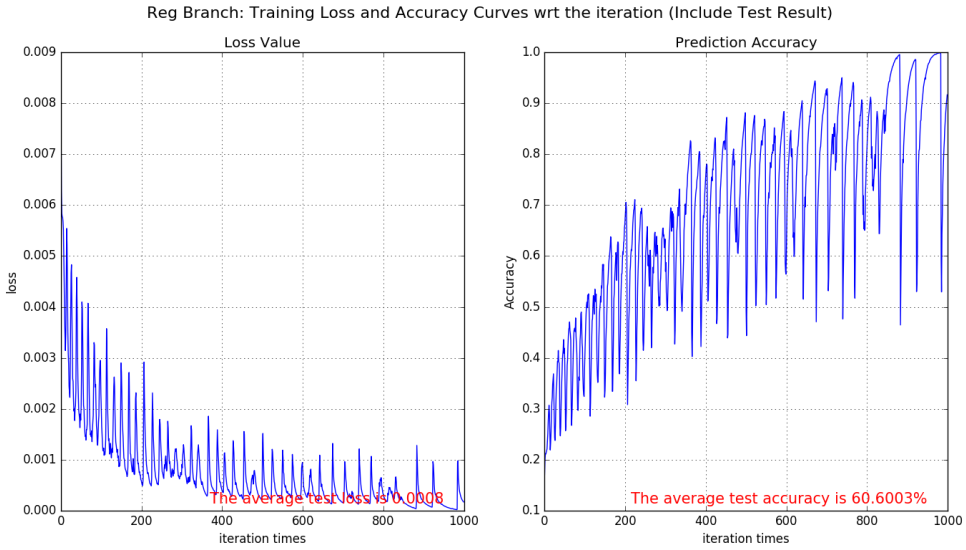

Figure 11: Training loss and accuracy curves (Proposal Regression Branch)

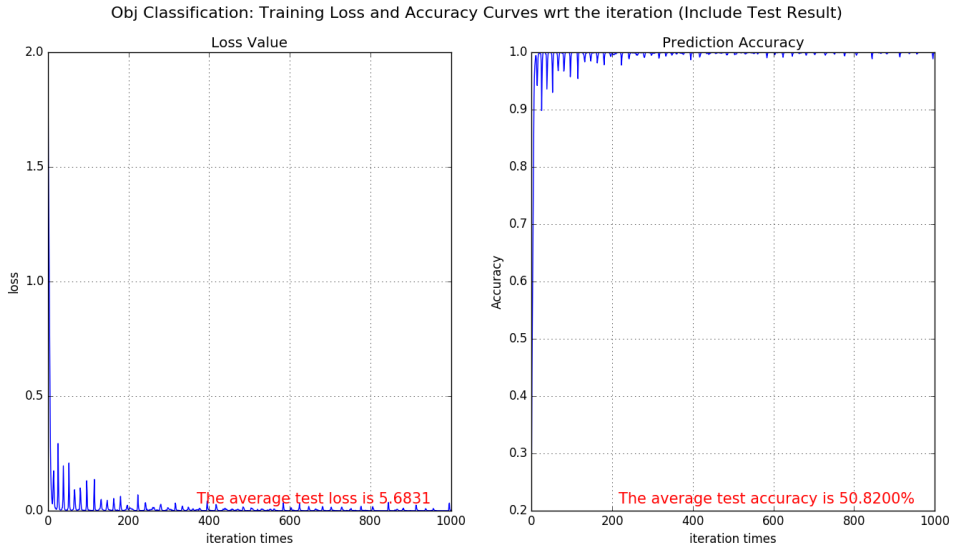

Figure 12: Training loss and accuracy curves (Object Classifier Branch)

According to the three curves above, all losses will decrease with the steps of training, however, there exists a lot of fluctuations especially in proposal regression branch. Here, I set different weights for three losses, cross entropy loss from proposal classifier branch, smooth l1 loss from proposal regression branch and final softmax loss from the object classification side. In a word, the whole tendency looks reasonable which proves the accuracy of model, and the final test accuracy can reach 50%.

3.3 Faster RCNN with MobileNet and ResNet Base

In this section, I replace the first three ConvNet in BaseNet with NobileNet and ResNet respectively, follow the similar training strategy as the previous one, get the results as below.

Below three figures show the training process and final test accuracy obtained from **MobileNet Based Structure**.

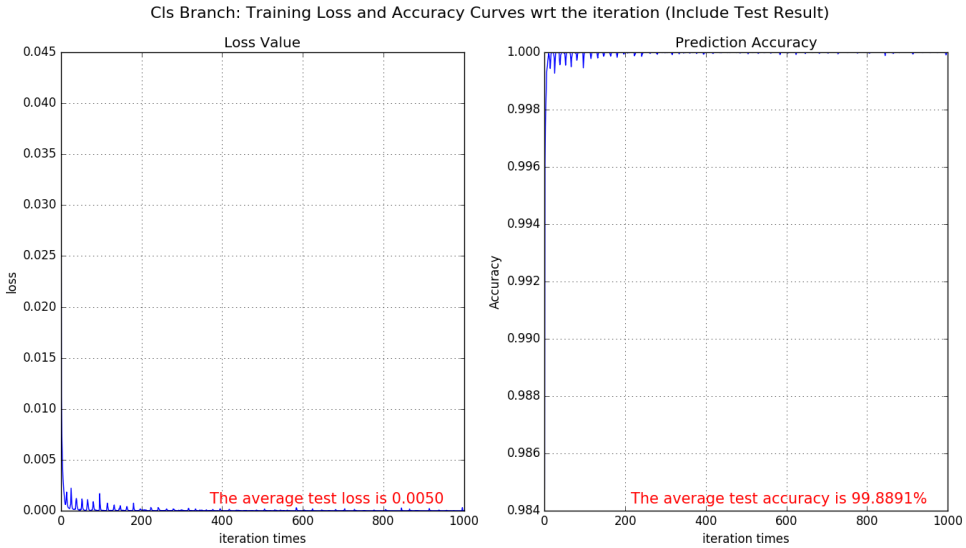

Figure 13: Training loss and accuracy curves (Proposal Classifier Branch - MobileNet)

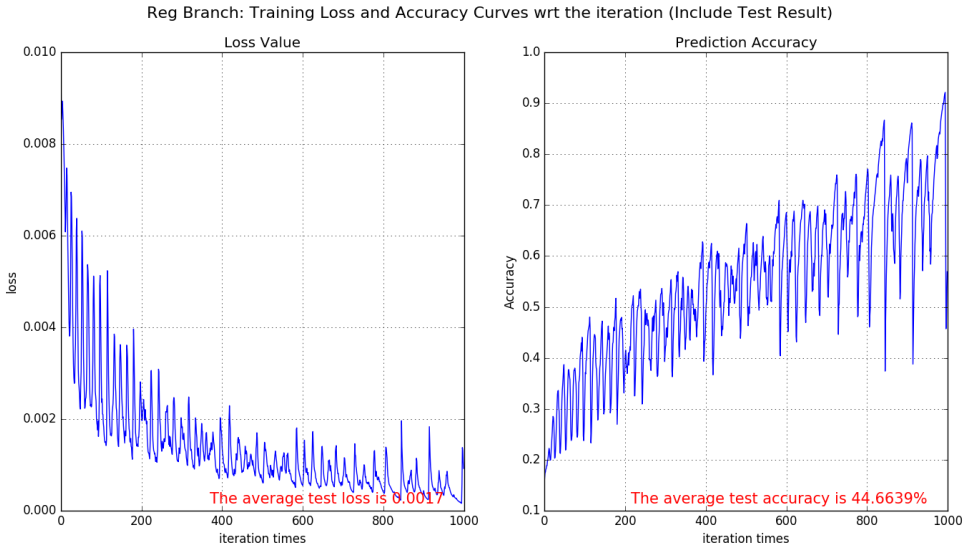

Figure 14: Training loss and accuracy curves (Proposal Regression Branch - MobileNet)

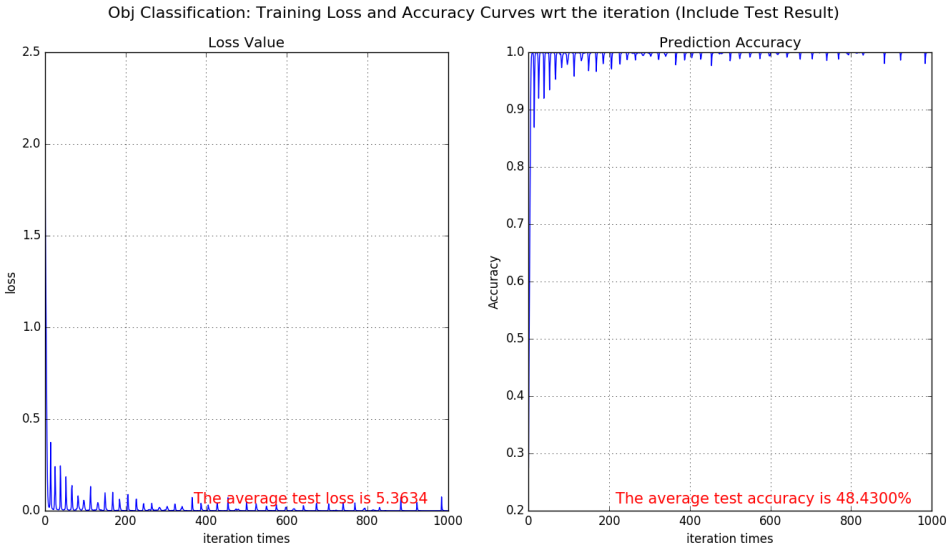

Figure 15: Training loss and accuracy curves (Object Classifier Branch - MobileNet)

Below three figures show the training process and final test accuracy obtained from **ResNet Based Structure**.

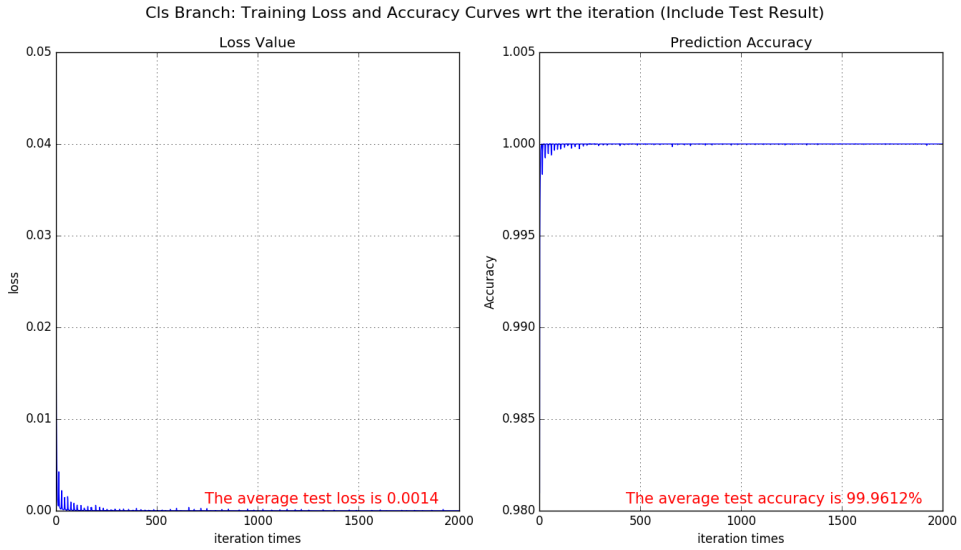

Figure 16: Training loss and accuracy curves (Proposal Classifier Branch - ResNet)

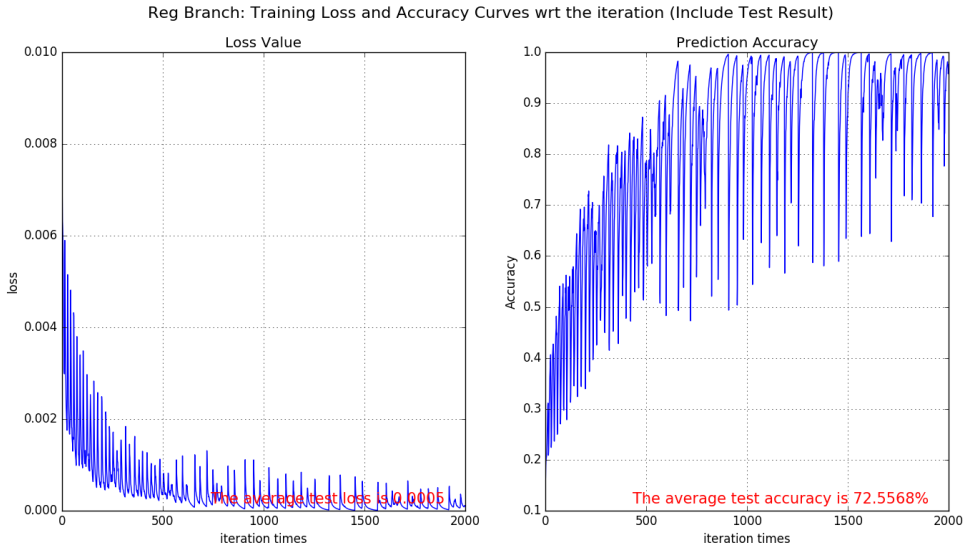

Figure 17: Training loss and accuracy curves (Proposal Regression Branch - ResNet)

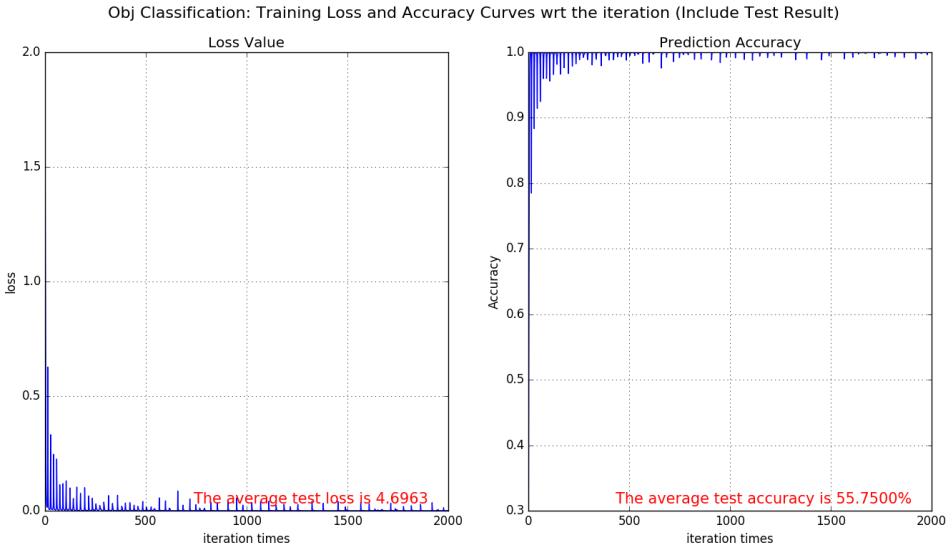

Figure 18: Training loss and accuracy curves (Object Classifier Branch - ResNet)

After the comparison of three base nets, it's obvious that the **ResNet Based Structure** can provide the best test accuracy in all three branches (99.9612% in cls branch, 72.5568% in reg branch, 55.75% in obj branch). However, due to its relative complicated structure in each single block, it almost costs double times for training compared with **ConvNet Based Structure**. Meanwhile, **MobileNet Based one** performs the worst among three base nets, and it also needs more time than the ConvNet so that is quite low efficient.

In addition, with different loss weight combinations and introducing drop-out layers can make the network performance better.