Homework 3

Name: Ty Nguyen
ID: tynguyen

1. Base Networks: ConvNet, MobileNet, and ResNet (30%)

Before deep learning emerges, computer vision researchers design features to describe and/or discriminate images, e.g., HOG, SIFT, Gist, etc. Nowadays, researchers turn to design deep architectures to learn features that are useful for various tasks including ob- ject recognition, object detection, and semantic segmentation. In this part, you will build three base networks and compare their performance and efficiency using the CIFAR-10 dataset.

1.1. (5%) Train a network with architecture shown in Table 1.

All convolution should preserve the input resolution with stride 1. Use Adam opti- mizer and exponential learning rate decay from 10 −3 to 10 −4 in 2, 000 iterations. Use weight decay 0.05. Report the final test accuracy and total training time. Count the total number of multiplications in convolution layers. (Do not count batch normalization.)

Final Test Accuracy: 64.6%

Training Time: 309 seconds

Total number of multiplications in convolution layers:

- conv 1, K, M, N, DF = 5 3 32 32, compute 2457600
- conv 2, K, M, N, DF = 5 32 64 16, compute 13107200
- conv 3, K, M, N, DF = 5 64 128 8, compute 13107200
- conv 4, K, M, N, DF = 5 128 256 4, compute 13107200
- conv 5, K, M, N, DF = 3 256 512 2, compute 4718592
- Total Multiplications: 46497792

In [1]:

```
import skimage.io as io
import matplotlib.pyplot as plt
```

In [2]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part1/convnet/accuracy.png')
fig = plt.figure(figsize=(8,7))
plt.imshow(img)
plt.axis('off')
plt.show()
```

1.2. (10%) Replace the first 4 convolution layers with the separable convolution layers as shown in Figure 1.

Final Test Accuracy: 51.6%

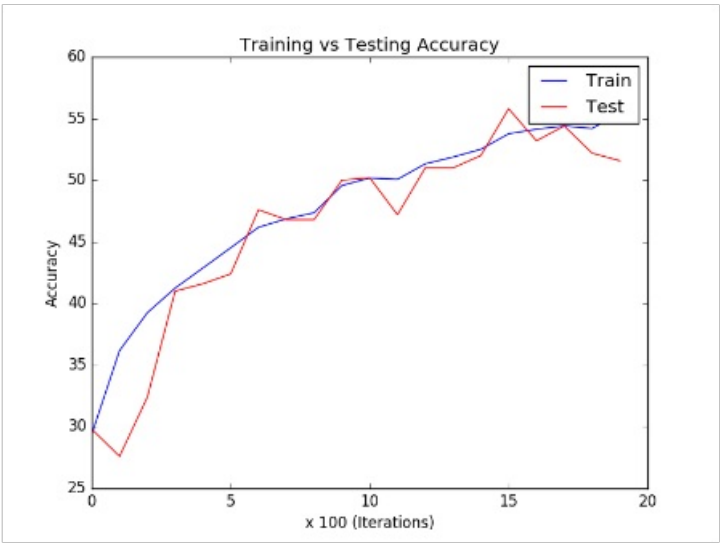
Training Time: 312 seconds

Total number of multiplications in convolution layers:

- conv 1, K, M, N, DF = 5 3 32 32, compute 175104
- conv 2, K, M, N, DF = 5 32 64 16, compute 729088
- conv 3, K, M, N, DF = 5 64 128 8, compute 626688
- conv 4, K, M, N, DF = 5 128 256 4, compute 575488
- conv 5, K, M, N, DF = 3 256 512 2, compute 4718592
- Total Multiplications: 6824960

In [3]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part1/mobilenet/accuracy.png')
fig = plt.figure(figsize=(8,7))
plt.imshow(img)
plt.axis('off')
plt.show()
```



1.3. (10%) Replace the first 4 convolution layers with the residual blocks as shown in Figure 2.

Final Test Accuracy: 68.4%

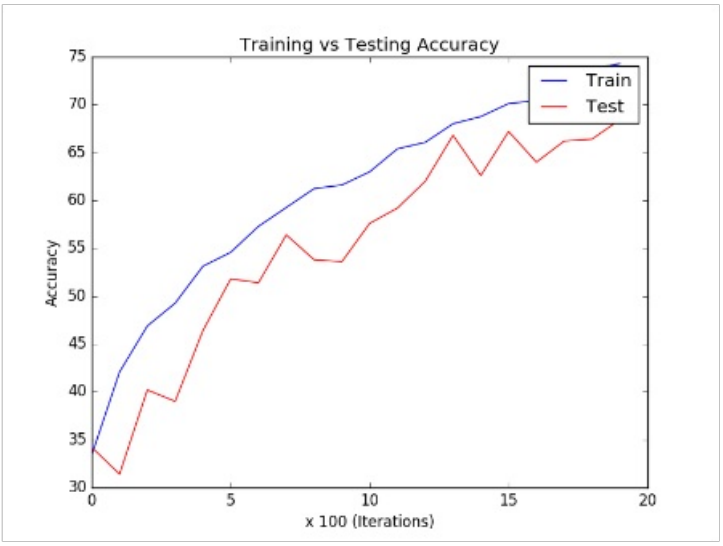
Training Time: 320 seconds (GPU)

Total number of multiplications in convolution layers:

- *Resblock conv 1, K, M, N, DF = 3 3 32 32, compute 138240
- *Resblock conv 2, K, M, N, DF = 3 32 32 32, compute 9437184
- *Resblock conv 3, K, M, N, DF = 1 3 32 32, compute 98304
- *Resblock conv 4, K, M, N, DF = 3 32 64 16, compute 368640
- *Resblock conv 5, K, M, N, DF = 3 64 64 16, compute 9437184
- *Resblock conv 6, K, M, N, DF = 1 32 64 16, compute 524288
- *Resblock conv 7, K, M, N, DF = 3 64 128 8, compute 184320
- *Resblock conv 8, K, M, N, DF = 3 128 128 8, compute 9437184
- *Resblock conv 9, K, M, N, DF = 1 64 128 8, compute 524288
- *Resblock conv 10, K, M, N, DF = 3 128 256 4, compute 92160
- *Resblock conv 11, K, M, N, DF = 3 256 256 4, compute 9437184
- *Resblock conv 12, K, M, N, DF = 1 128 256 4, compute 524288
- conv 13, DK, M, N, DF = 3 256 512 2, compute 4718592
- Total Multiplications: 44921856

In [4]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part1/resnet/accuracy.png')
fig = plt.figure(figsize=(8,7))
plt.imshow(img)
plt.axis('off')
plt.show()
```



1.4. (5%) Compare and explain the results in the previous questions.

As we can see from the above results:

- Number of multiplicationd: the largest is Resnet and the smallest is Mobilenet. Replacing standard convolutional layers by separable convolution layers and residual blocks in this situation help reduces a large number of computations. This can be seen from the equation computing the number of multiplications in each convolutional layer:

  $$muls = K^2 * M * N * D_F^2$$

  where K is the kernel size (assumed to be square), D_F is the width/height of the output feature, M, N are the number of input features and outputs features, respectively. The separable convolutional layers break down the multiplication on the right side of this equation into two operhands whose sum is by far smaller:

  $$muls = K^2 * M * D_F^2 + M * N * D_F^2$$

  , thanks to the combination of depthwise convolution and 1 × 1 (pointwise) convolution. The residual block also helps reduce the number of computations by using a set of convolutional layers with smaller kernels.
- In running time: although there is a big different in the number of multiplications required by the Convnet and Mobilenet, the running time of them is almost identical since I used GPU. An experiment in CPU shows that the Resnet and Convnet is much slower than Mobilenet.
- In accuracy: resnet outperforms the other networks. This can be explained by the fact that the skip connection allows it to go deeper without much overfitting.

2. Region Proposal Network (40%)

In this part, you will build a region proposal network step by step with the base networks you developed in the previous part. Region proposal network is a part of Faster R-CNN (Figure 3) for localizing objects in an image.

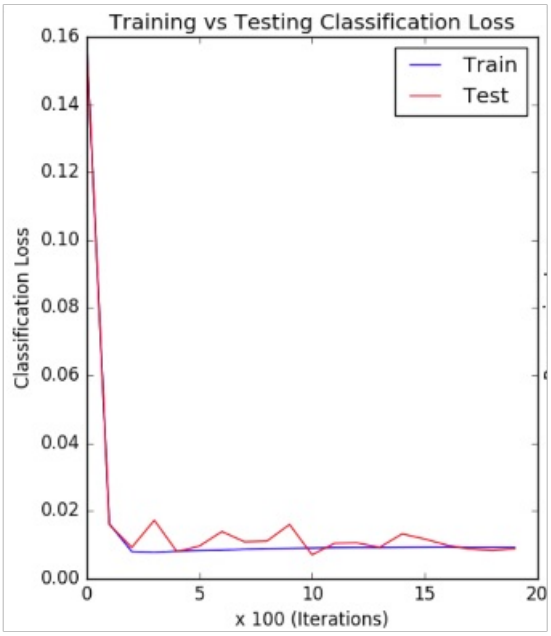
2.1. (20%) Build a base network as shown in Table 2.

ReLU activations, pooling and softmax layers. (No fully connected layer except for softmax.) The test accuracy has to reach 68% with the same training procedure. Feel free to use any image pre-processing or augmentation methods. Provide a table that describes the network architecture. Plot the training accuracy over training iterations and report the final test accuracy.

Training accuracy

In [45]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part2/2.1/class_loss.png')
fig = plt.figure(figsize=(8,7))
plt.imshow(img)
plt.axis('off')
plt.show()
```



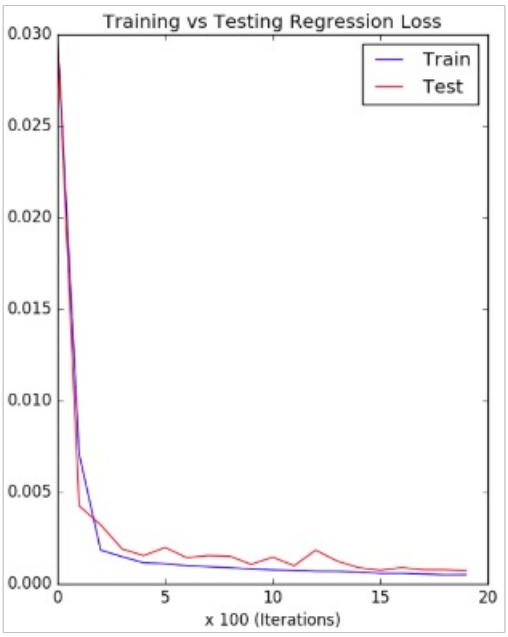
(point-wise) Test Accuracy of the proposal classifier: 0.0101

2.2. (20%) In this question, you will build a proposal regressor (as the reg branch in Figure 3).

Training Regression Loss

In [17]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part2/2.2/regression_loss.png')
fig = plt.figure(figsize=(8,7))
plt.imshow(img)
plt.axis('off')
plt.show()
```



Final testing regression loss: 0.000727

3. Faster R-CNN (30%)

With the base network and region proposal network ready, you are now only one step from completing Faster R-CNN. After finishing the whole pipeline of Faster R-CNN, you can also change the base network to see the effect of different learned features.

3.1. (10%) One important layer that transforms the features of the proposal into the final object classifier is the ROI pooling layer.

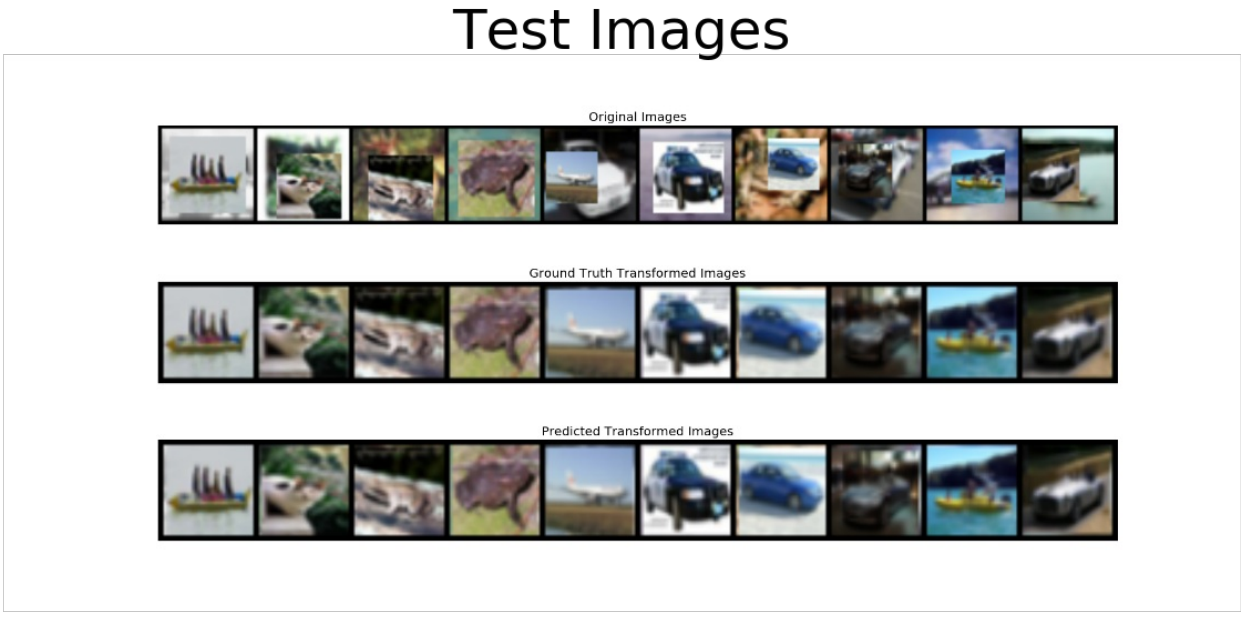
The ROI pooling layer warps the features of an ROI region into a fixed-sized feature map. Display few outputs of the spatial transformer layer as shown in Figure 5

In [39]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part2/2.2/train_transformed_images.png')
fig = plt.figure(figsize=(20,20))
plt.imshow(img)
plt.axis('off')
plt.title("Training Images", fontsize=50)
plt.show()
```

In [38]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part2/2.2/test_transformed_images.png')
fig = plt.figure(figsize=(20,20))
plt.imshow(img)
plt.axis('off')
plt.title("Test Images", fontsize=50)
plt.show()
```



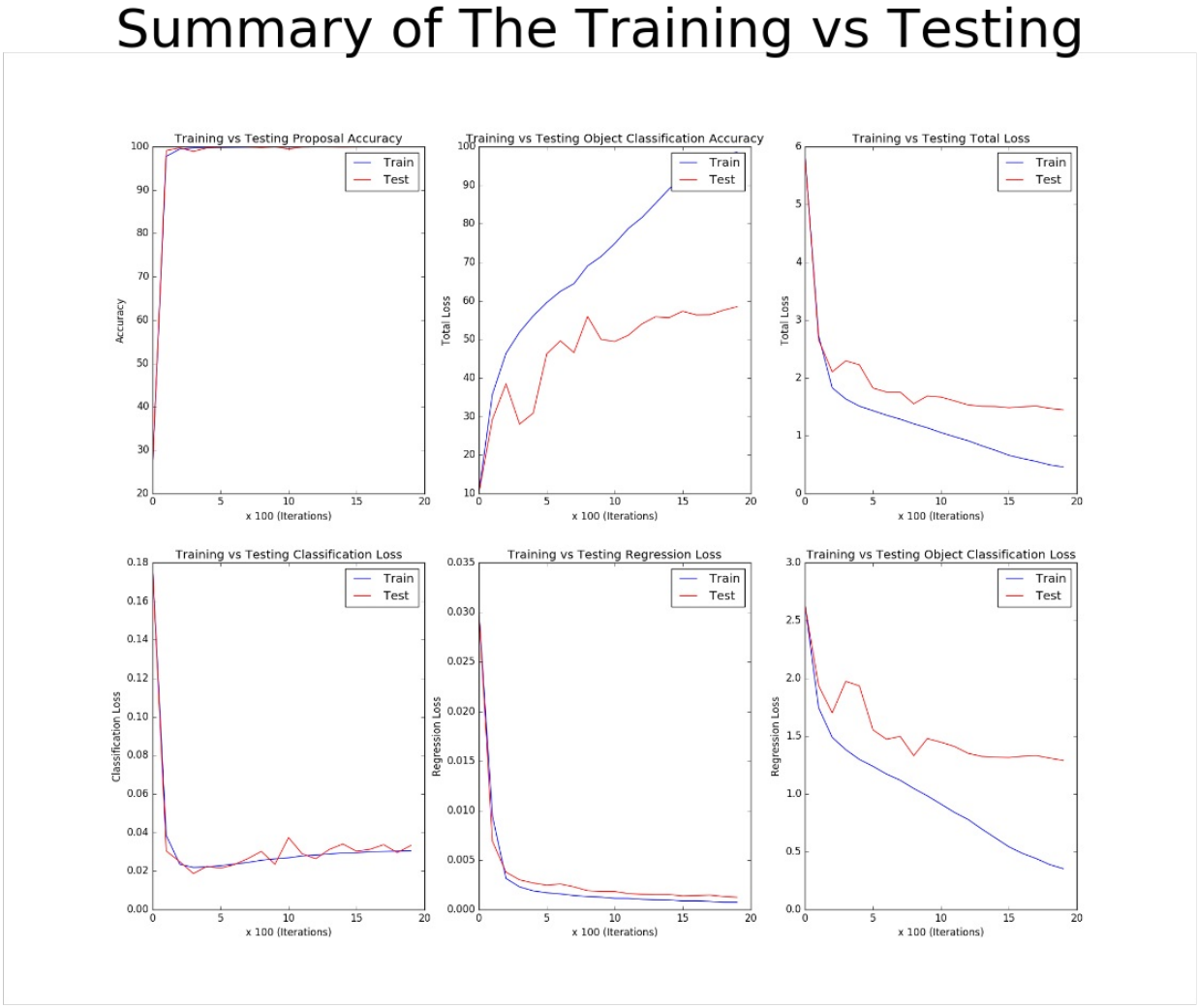
3.2 (10%) Use the same θ parameter for the spatial transformer layer; however, feed in

the feature map (conv4) from the base network and set the output size as (4, 4). Add a fully convolution layer with 256 output channels (followed by BatchNorm and ReLU) and a 10-way Softmax layer (with fully connected layer) for classification. Train the network (consisting of proposal classifier, regressor, and object classifier) end-to-end with the same training procedure.

Plot the 3 training losses over training iterations. Report the final classification test accuracy.

In [40]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part3/3.2/simple_loss/fasterrcnnnet/total_loss.png')
fig = plt.figure(figsize=(20,20))
plt.imshow(img)
plt.axis('off')
plt.title("Summary of The Training vs Testing", fontsize=50)
plt.show()
```



Report the final classification test accuracy: 58.46%

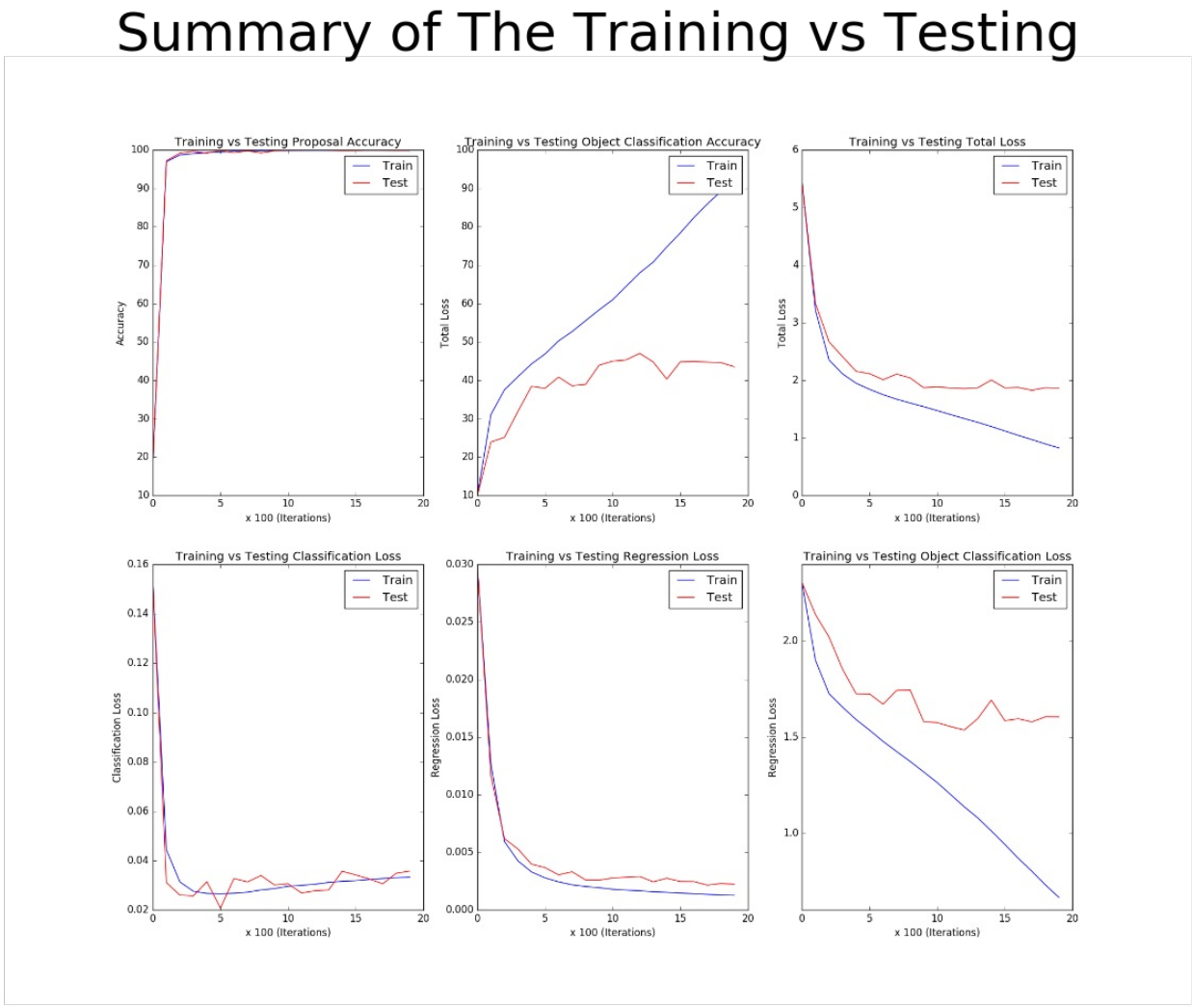
3.3 (10%) Replace the first three convolution layers in the base network (Table 2)

with separable convolution layers or residual block respectively. Repeat the experiments in the previous question.

Plot the 3 training losses over training iterations for Mobile Base network.

In [50]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part3/3.2/simple_loss_mobile_net/fasterrcnnmobilenet/object_
loss.png')
fig = plt.figure(figsize=(20,20))
plt.imshow(img)
plt.axis('off')
plt.title("Summary of The Training vs Testing", fontsize=50)
plt.show()
```

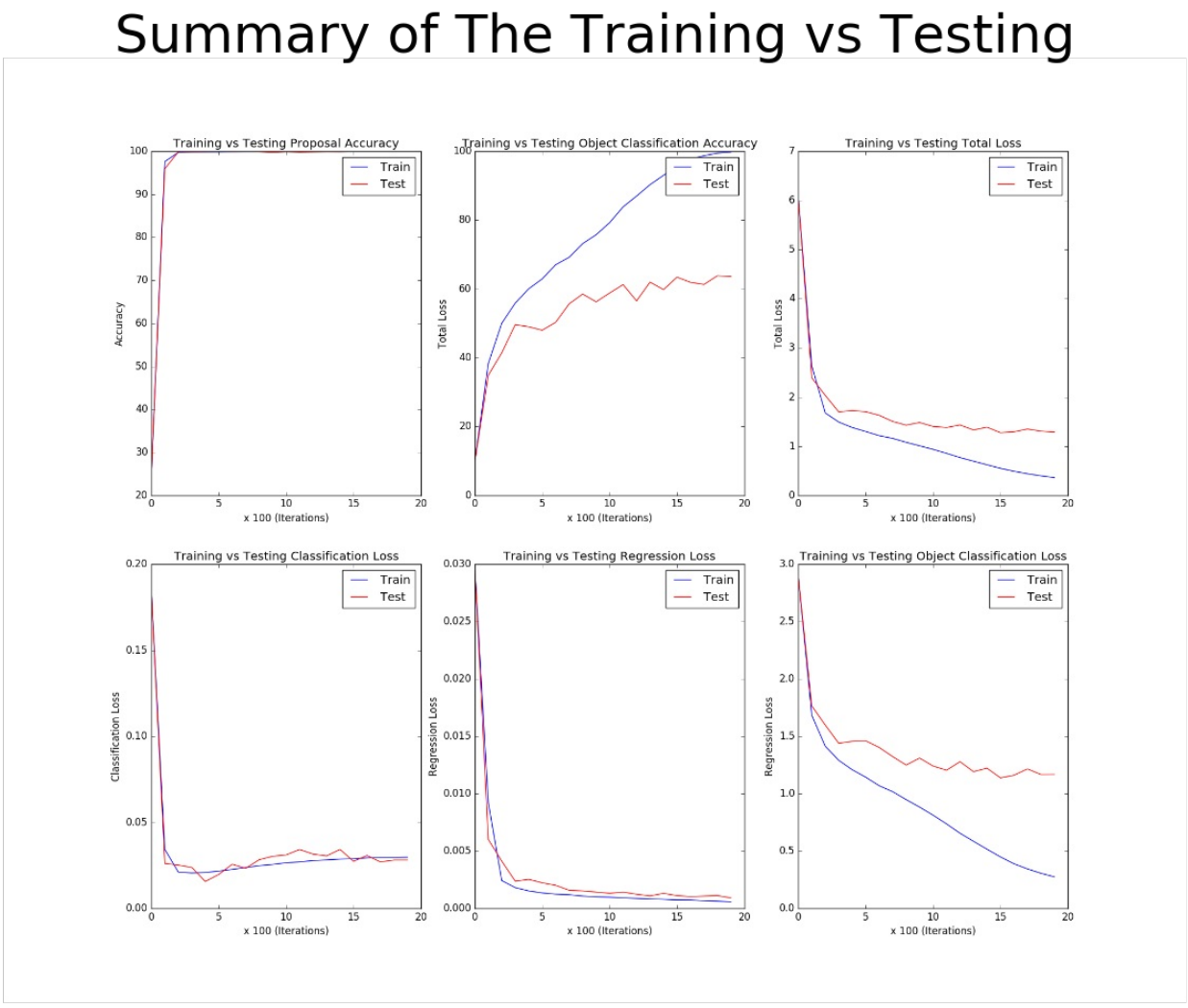


Report the final classification test accuracy: 43.53%

Plot the 3 training losses over training iterations for Residual Base network.

In [42]:

```
img = io.imread('/home/tynguyen/cis680/logs/HW3/part3/3.2/simple_loss_res_net/fasterrcnnresnet/object_loss.png')
fig = plt.figure(figsize=(20,20))
plt.imshow(img)
plt.axis('off')
plt.title("Summary of The Training vs Testing", fontsize=50)
plt.show()
```



Report the final classification test accuracy: 63.66%

Some observations:

The proposal classification and bounding box regression parts converse much faster than the object classification part.

Using alternative training approach is not considerably better than using a single training phase with a proper set of weights for different losses.

Setting weights for losses so that they have a approximately equal magnitude value in the total loss does not help but may harm the performance. In this case, the initial object classification loss is 2.54 which is much greater than others (~0.2 and 0.02) but a set of weights (1, 100, 1) provides the best result.

Compare the results of Faster R-CNN with all three base networks: using the same parameters, residual base network provides the best performance while the mobile base network is inferior to the others. This observation is consistent with the results given in part 1 where Resnet is also superior to the others.

During the training, I used the ground truth rather than the predicted bounding box to obtain theta used to transform the features from Conv4 layers. This is shown to help increase the accuracy.

There is a overfitting occuring the object classification part which consists of two fully-connected layers. This overfitting exists consistently in all three networks can be explained by the fact that we do not use drop-out or any other regularizations. Thus, incoporating a regularization can also help improve the performance.