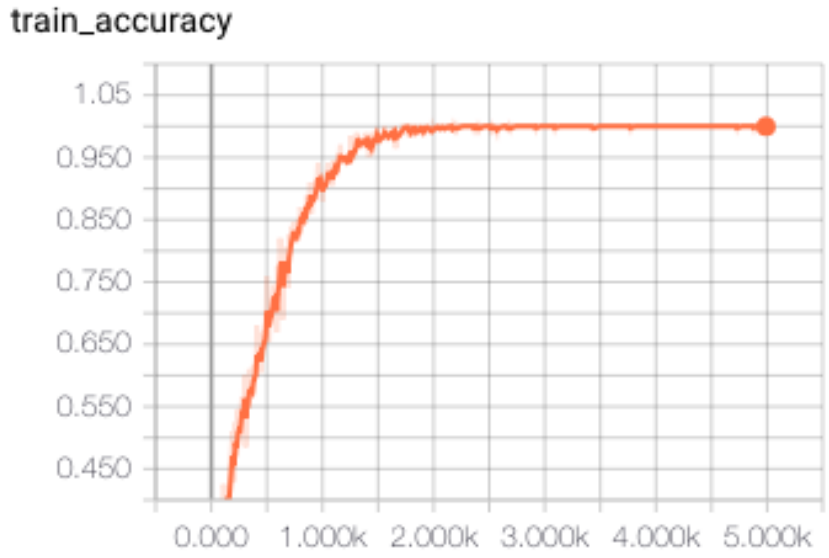
Deep Learning Assignment 2

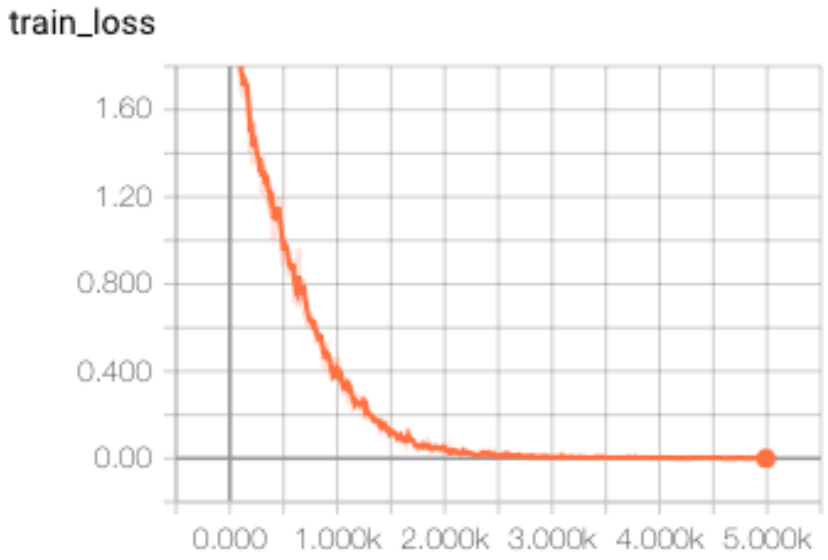
arc11

October 2018

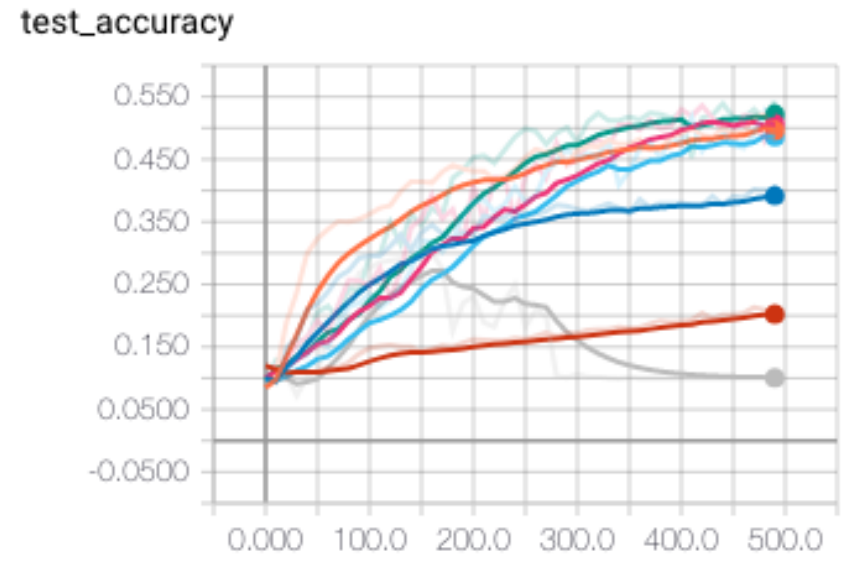
1 Problem 1

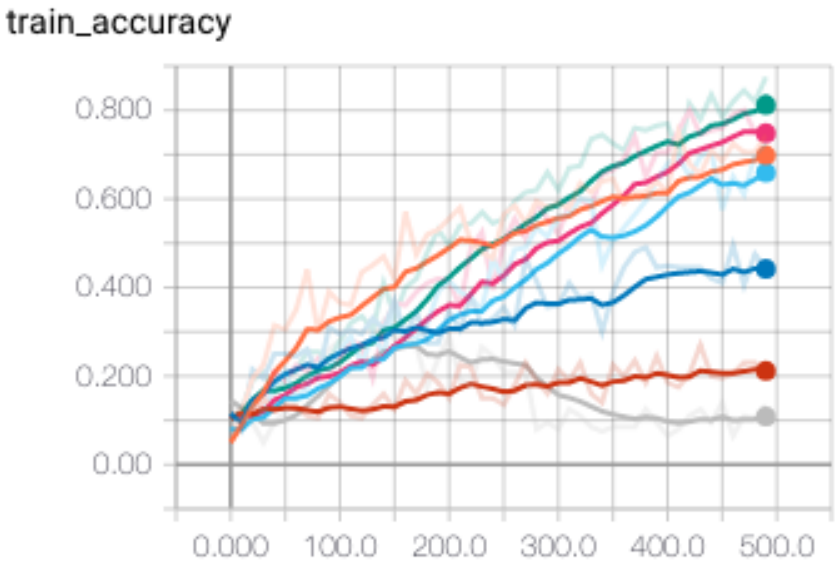
1.1 Train and Test Accuracy and Train Loss Curves

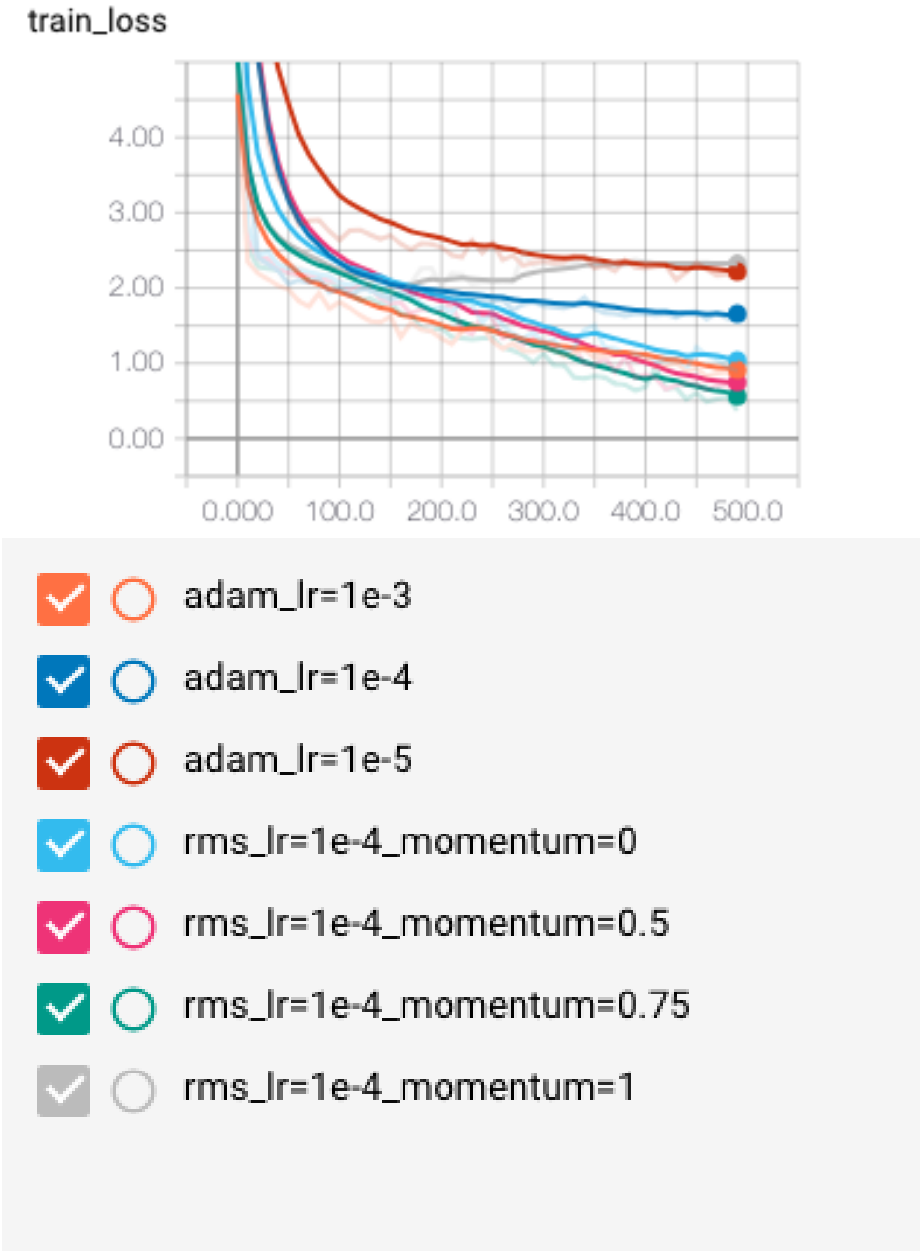



train_loss
1.60
1.20
0.800
0.400
0.00
0.000
1.000k
2.000k
3.000k
4.000k
5.000k

1.2 Testing with different hyperparameters

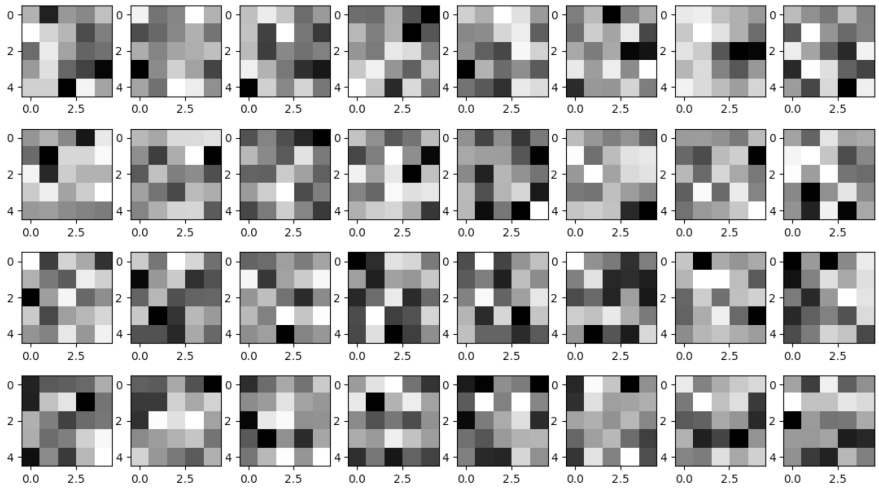






Looking at only the results of the ADAM learning algorithm, you can see that, as expected, higher learning rates correlate with faster learning. For RMSProp, I kept the learning rate constant and adjusted the momentum value and saw that 0 and 1 performed terribly, but 0.5 and 0.75 performed equal to or better than a higher learning rate on ADAM.

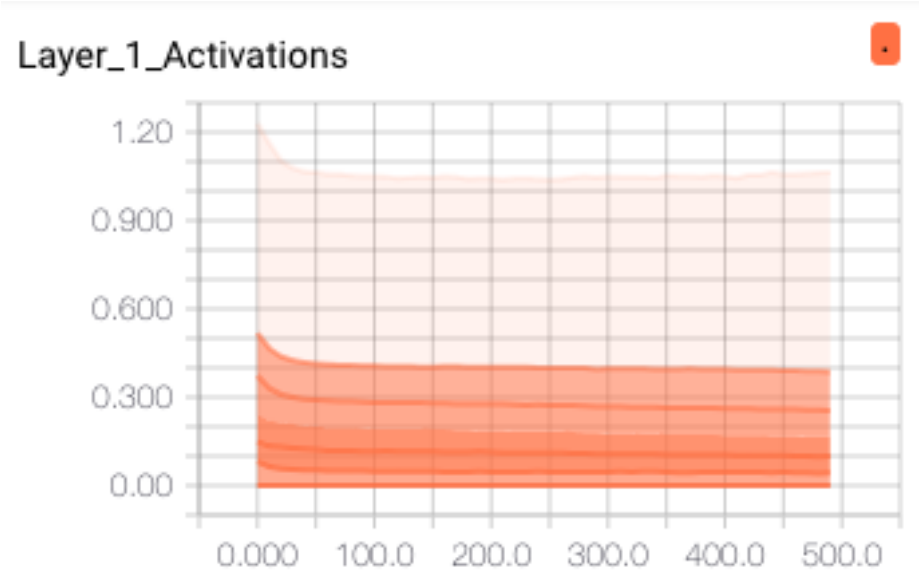
1.3 First Convolutional Layer Weights Visualization (Gabor Filters)

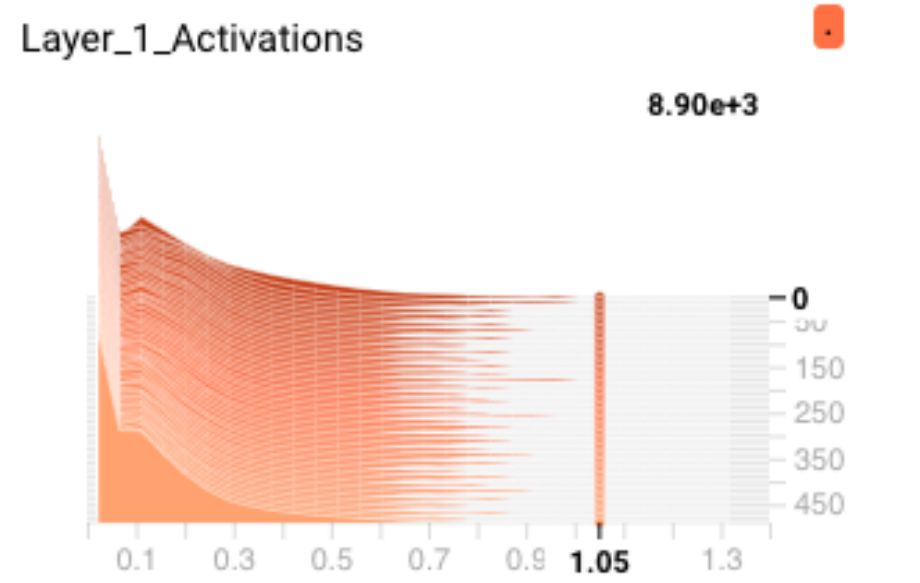


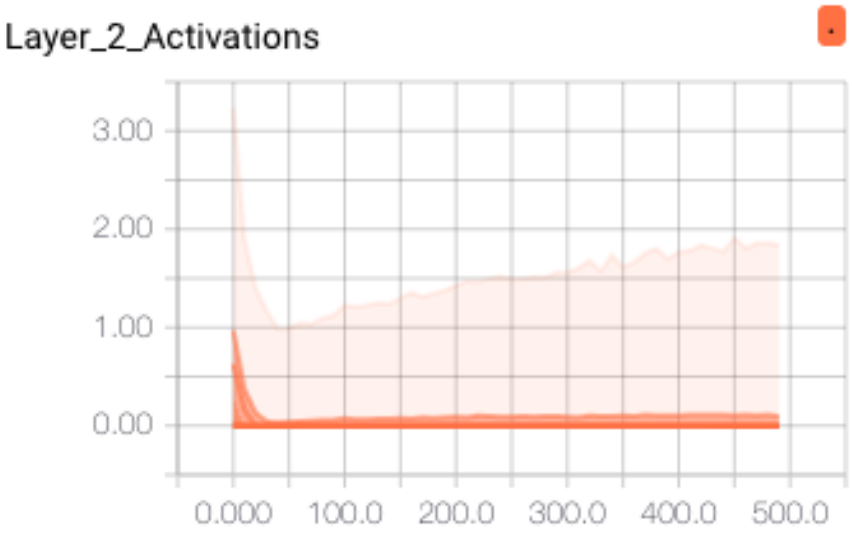
You can see that, although it is difficult to tell with only a single color channel, the weights do look slightly like Gabor filters. The lower left filter looks like it is selecting round shapes while some others look like they are doing straight edge detection.

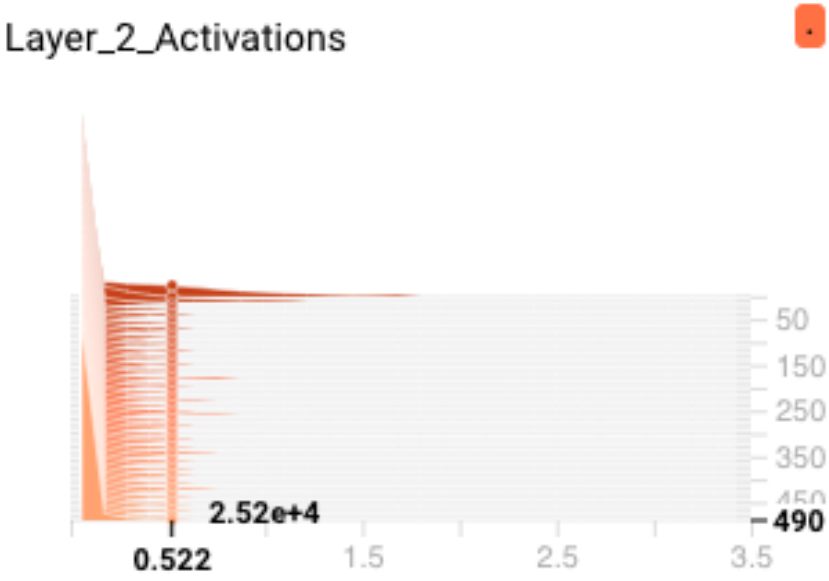
1.4 Activation Statistics

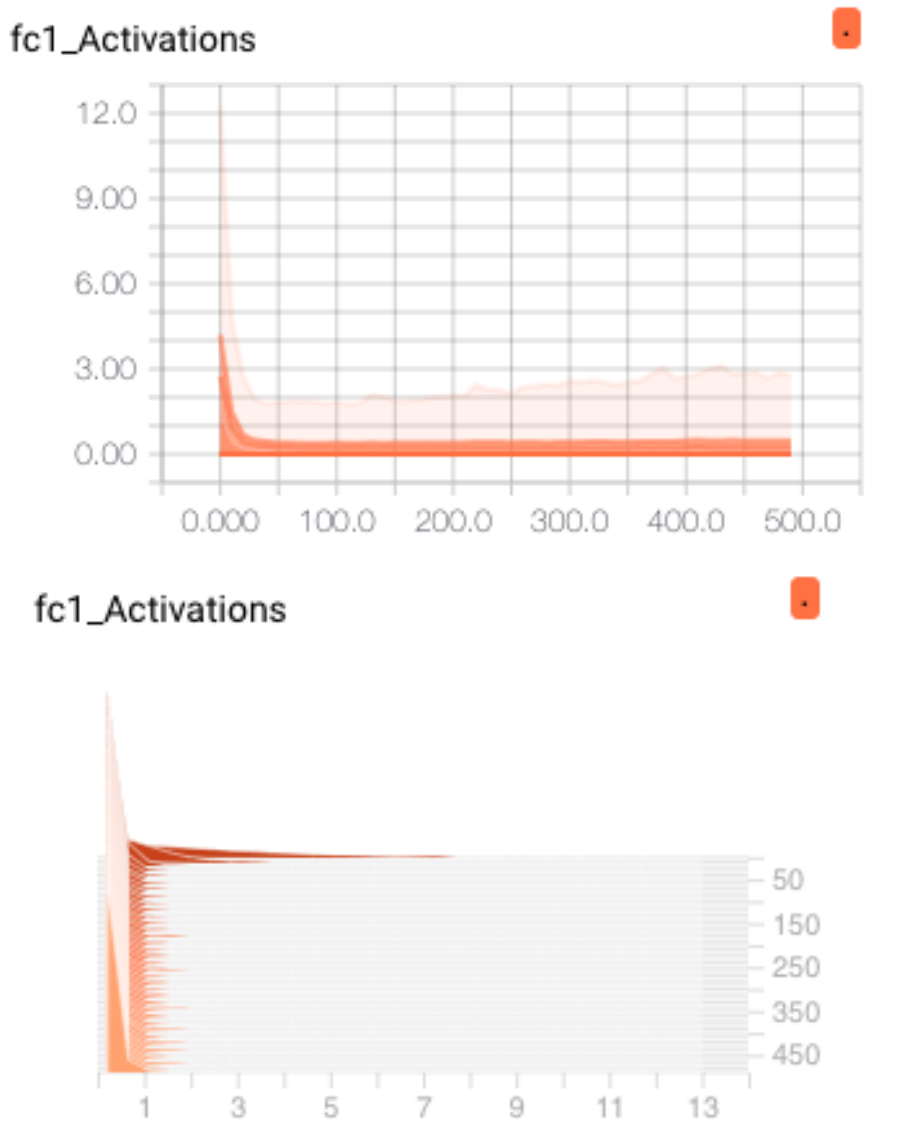
Below are the statistics of the activations on test images from different layers of the network. You can see that the early layers converge very quickly while the fully connected layers take longer to adjust.

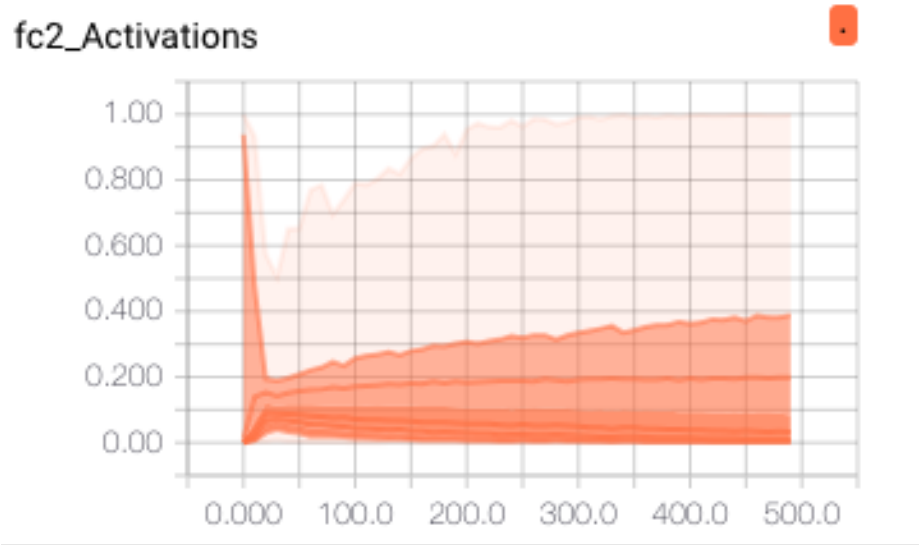

Layer_1_Activations
1.20
0.900
0.600
0.300
0.00
0.000
100.0
200.0
300.0
400.0
500.0

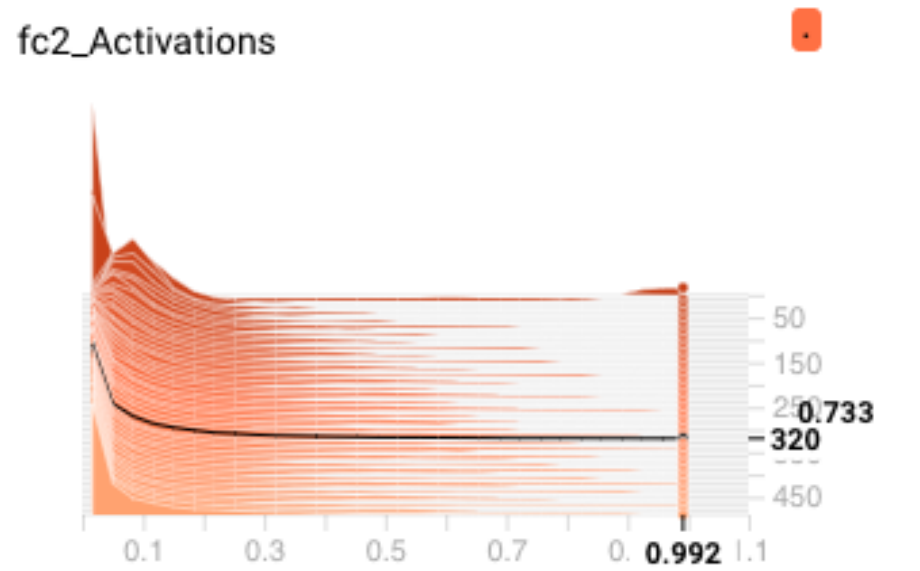

Layer_1_Activations
8.90e+3
0
150
250
350
450
0.1
0.3
0.5
0.7
0.9
1.05
1.3


Layer_2_Activations
3.00
2.00
1.00
0.00
0.000
100.0
200.0
300.0
400.0
500.0


Layer_2_Activations
50
150
250
350
2.52e+4
490
0.522
1.5
2.5
3.5


fc1_Activations
12.0
9.00
6.00
3.00
0.00
0.000
100.0
200.0
300.0
400.0
500.0
fc1_Activations
50
150
250
350
450
1
3
5
7
9
11
13





2 Problem 2

Although it is easy to see the effects of the inputs from the first convolutional layer, that is not the case for higher convolutional layers because those layers are not a direct linear transformation of the input space. This paper used a method called deconvolutional neural networks to map features from different layers in the convnet back to the pixel space in order to understand the patterns that created those weights. In order to examine an activation, they first feed the

image into the convnet, then they take an activation about and zero out all of the other activations in that layer. Then they pass that into the deconv that is attached to the layer, unpool, rectify, and then filter to extract a reconstruction of the input that caused that specific activation.

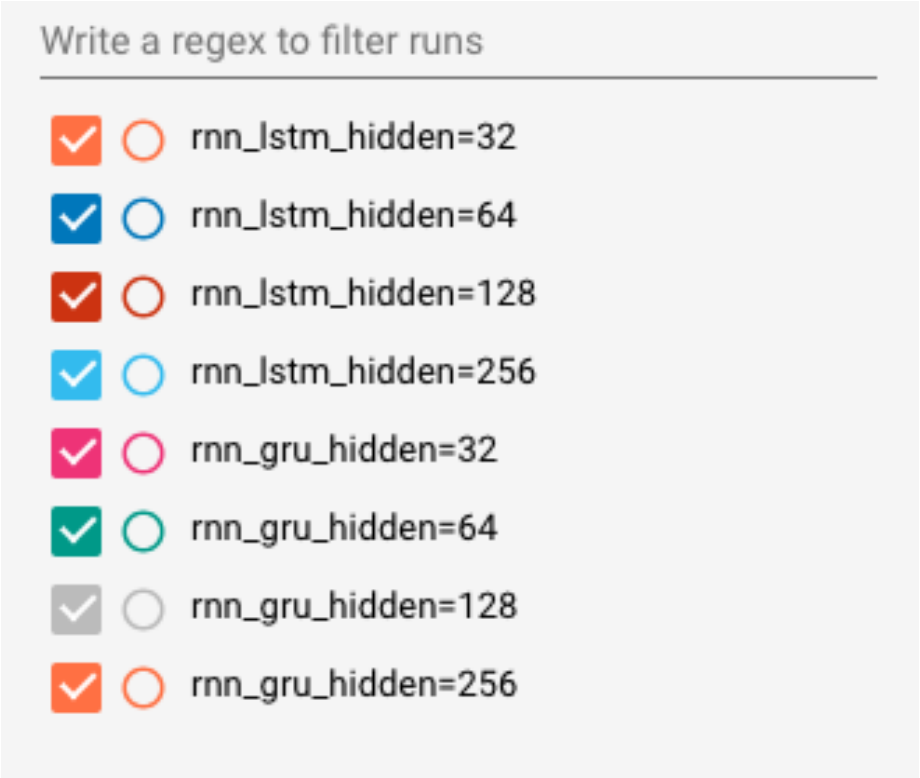
For a specific image, they are able to look at the highest activations and then determine the pattern that specific high activation is recognizing. Using this method you are able to project a feature onto the pixel level space and create an effective reconstruction of the input from a single activation. One key finding from this method is that the convnet creates a hierarchy of features that increases in complexity as you go to higher layers. Additionally, when analyzing how these feature maps evolved over training epochs, they found that feature maps associated with higher layers continually changed what they responded to throughout the training. This finding highlights the significance of training to convergence.

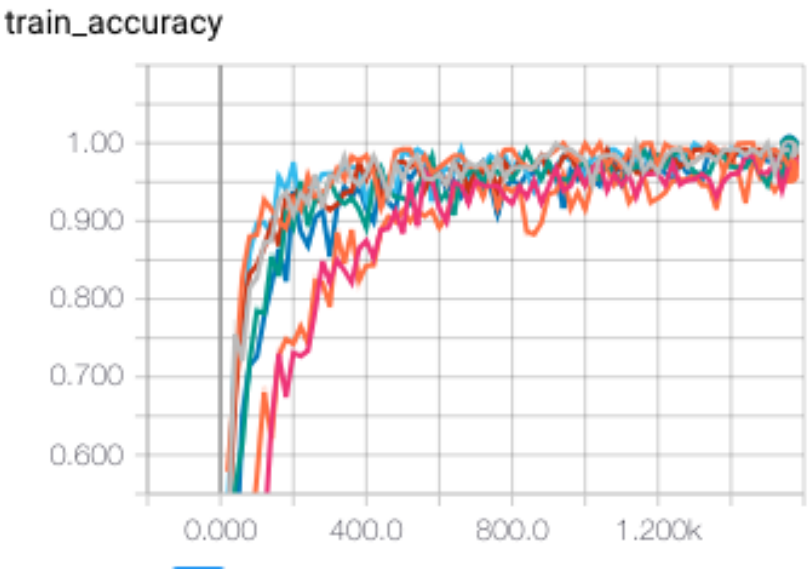
Using the deconvolution method, they were able to show that the neural network was gaining information from the actual object rather than the background of the object. They showed this by occluding parts of the image during inference and observing class changes as well as changes in the feature map's response to the stimulus. This paper also showed how shift, scale, and rotational invariance changes with the height of the layer. Lower layers are much more affected by these minor transformations of the input space. Another application of this method is to determine correspondences between parts of similar images. They did this by occluding some part of an image on similar classes and saw how that affected the feature map. In doing this, they are able to determine which layers pattern match to which features.

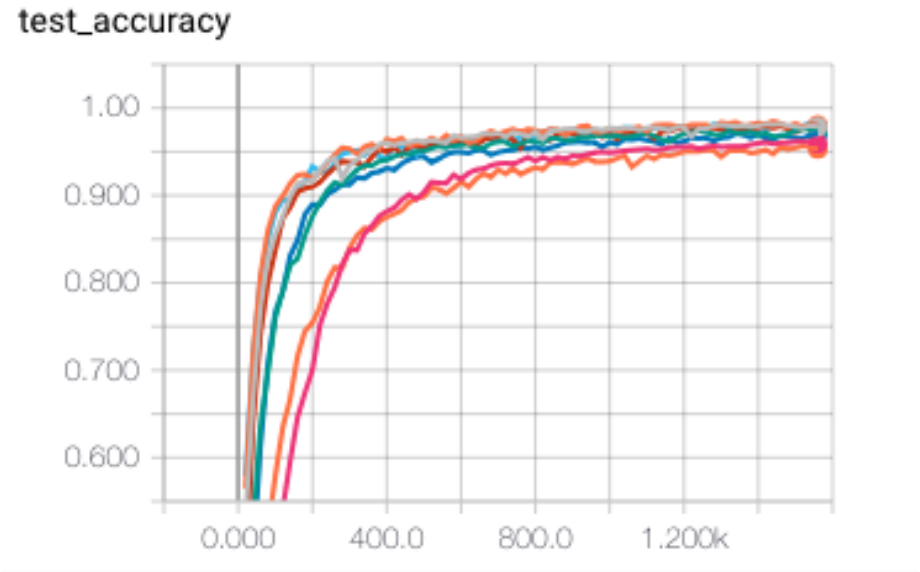
This method can also help select good architectures because it can show you what features are missing from your model. They compared the architecture they found by making improvements to the Krizhevsky using information from feature maps to other prominent architectures and found that it outperformed other architectures in those tasks.

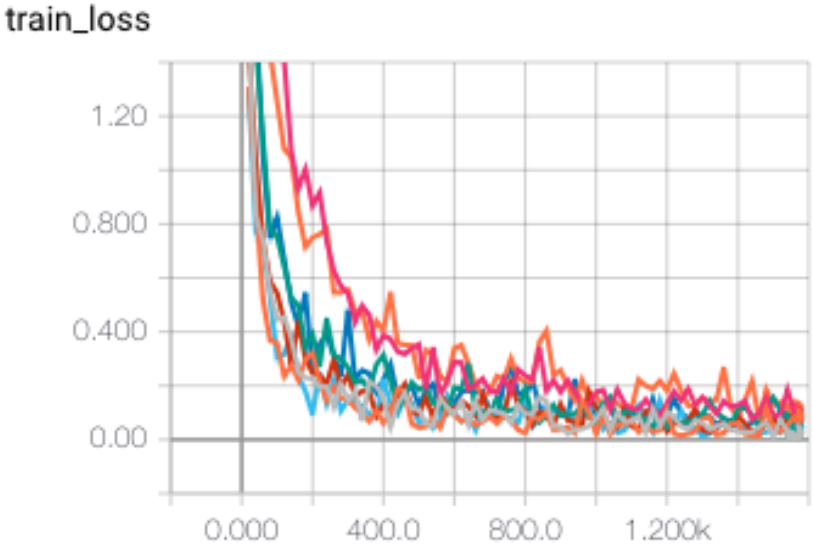
3 Problem 3

3.1 Testing Many Hyperparams









In every case, increasing the number of units in the hidden layer increased the training speed and accuracy. Additionally, GRU outperformed LSTM when you look within groups of the same n umber of hidden layers. However, the variation from GRU to LSTM was much smaller than the variation between number of hidden units.

3.2 Compare with CNN

I did not notice any significant differences between the RNN and CNN when training on CIFAR10. Looking at the test accuracy and training loss from the CNN, both networks get almost perfect accuracy at around 1000 iterations and the loss curves have the same structure, dipping below 0.2 at around 1000 iterations