1. The CIFAR-10, 10 class image data, is downloaded from the source: https://www.cs.toronto.edu/~kriz/cifar.html. In particular, the train data consists of 5 batches and each batch contains 10000 images of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, house, ship and truck and the testing file contains 10000 images.

   The data is loaded by *pickle* library, and the training labels (trY) in both train and test data is convert to an array size of 10 to present for actual label as MNIST DATA (1 at index 0 to 9, presenting for the number 0 to 9). As the result, there will be 10 output from the softmax output layer.
2. The experiment is done with 3 different convolution layers from 1 to 3. In particular, the filters size is used in the experiment is [3x3] and max_pooling with S = 2, F = 2. The train neural network using backpropagation with RMS optimizer with learning rate 0.001 and beta is 0.9 (just similar to the original activation function in the cnn.py file). Dropout strategy is only applied for the last convolutional layer and the fully connected layer.
3.

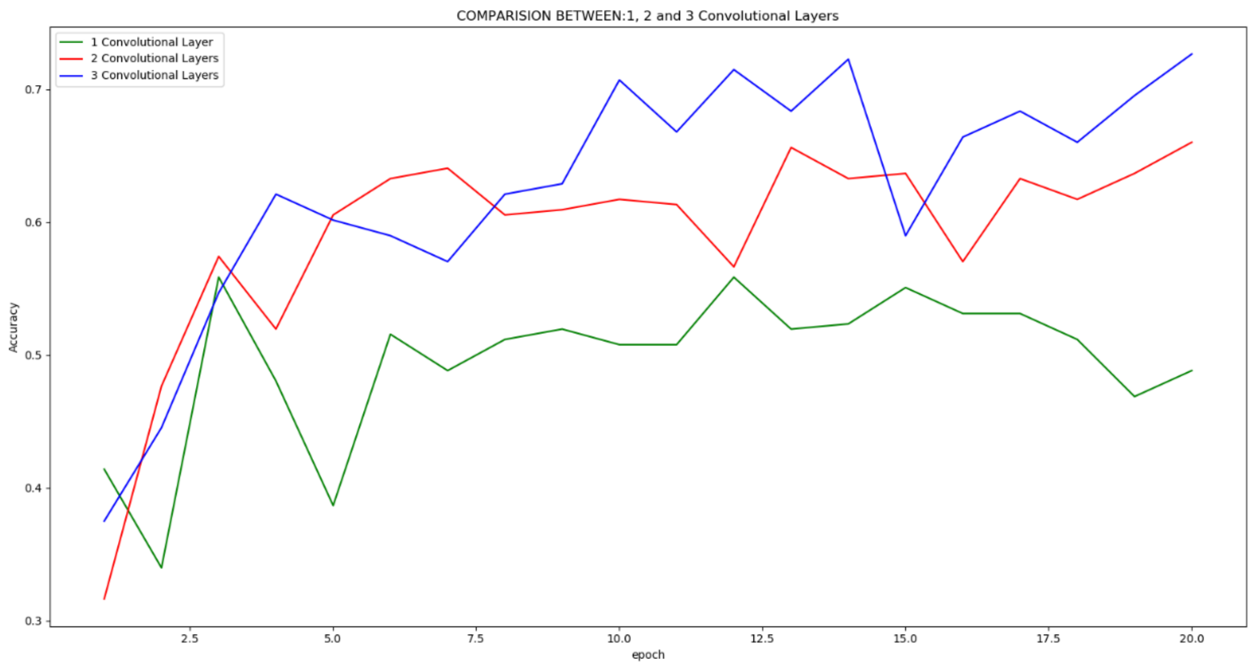
   a. For the 1-layer experiment, there are 6 filters used, then the shape of the first convolutional layer is [32x32x6] (padding = same is used), then max pool. The activation map shape would be [16x16x6] and it is used for the fully connected layer 300 outputs, then output result outs.

      For the 2 layers experiment, there are 6 filters for the first convolutional layer, then the shape of the first convolutional layer is [32x32x6] (padding = same is used), then max pool. Then the activation map shape would be [16x16x6]. For the second convolutional layer, there are 36 filters be used, with max pooling, as the result, the shape of the second convolutional layer will be [8x8x36]. Then, FCL the results.

      For the 3 layers experiment, there are 6 filters for the first convolutional layer be used. Then the shape of the first convolutional layer is [32x32x6] (padding = same is used), then max pool. Then the activation map shape would be [16x16x6]. For the second convolutional layer, there are 36 filters be used, with max pooling, as the result, the shape of the second convolutional layer will be [8x8x36]. For the third convolutional layer, there are 32 filters be used, with max pooling, therefore, the shape of the third convolutional layer will be [4x4x32]. Then, FCL and output the results.
      The summary of the experiment of 20 epochs is:

      The highest accuracy is 0.7265625 with the number of layers 3
      The average accuracy of 1 layer is 0.495703125
      The average accuracy of 2 layers is 0.591015625
      The average accuracy of 3 layers is 0.62578125

The highest accuracy is 0.727 and the winner is 3 layers, along with that, the average of these 3 layers can also indicates that the 3 layers performs better than 1 and 2 layers with the models described above.

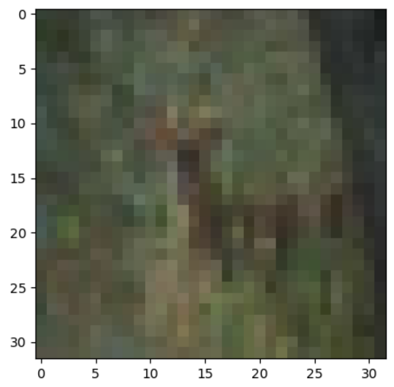


Here is the table of accuracy:

1 Layer	2 Layers	3 Layers
0 0.4140625	0 0.31640625	0 0.375
1 0.33984375	1 0.4765625	1 0.4453125
2 0.55859375	2 0.57421875	2 0.546875
3 0.48046875	3 0.51953125	3 0.62109375
4 0.38671875	4 0.60546875	4 0.6015625
5 0.515625	5 0.6328125	5 0.58984375
6 0.48828125	6 0.640625	6 0.5703125
7 0.51171875	7 0.60546875	7 0.62109375
8 0.51953125	8 0.609375	8 0.62890625
9 0.5078125	9 0.6171875	9 0.70703125
10 0.5078125	10 0.61328125	10 0.66796875
11 0.55859375	11 0.56640625	11 0.71484375
12 0.51953125	12 0.65625	12 0.68359375
13 0.5234375	13 0.6328125	13 0.72265625
14 0.55078125	14 0.63671875	14 0.58984375
15 0.53125	15 0.5703125	15 0.6640625
16 0.53125	16 0.6328125	16 0.68359375
17 0.51171875	17 0.6171875	17 0.66015625
18 0.46875	18 0.63671875	18 0.6953125
19 0.48828125	**19 0.66015625**	**19 0.7265625**

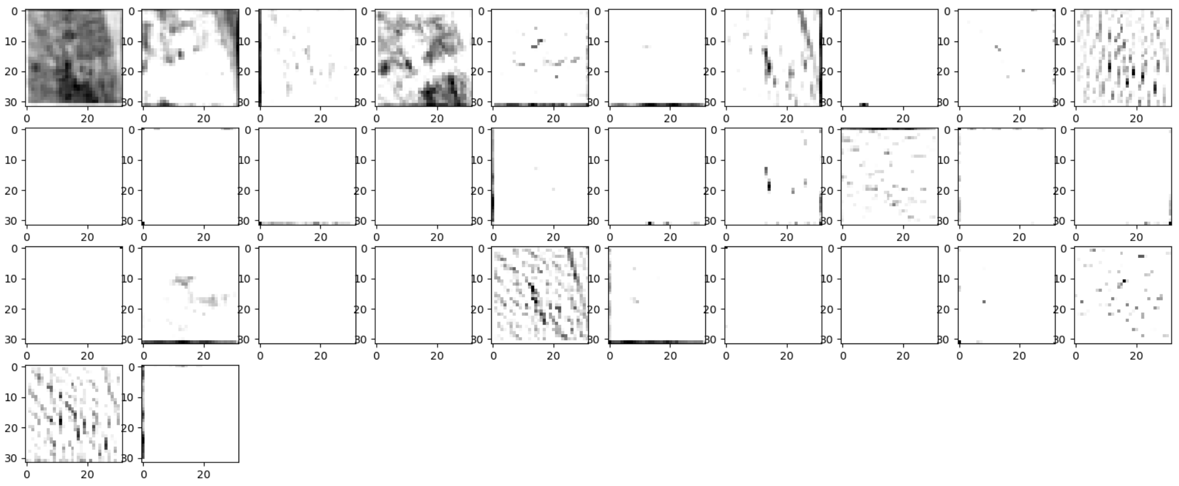
b. As the result from part a, I decide to modify the LeNet-5 model with 3 layers. Instead of using max pooling, I would use average pooling in this experiment and keep the size of the filters same as the previous experiment.
   The experiment is testing the accuracy of 3 different set of filters which is [6,16,32]; [16,32,64]; [32,64,128]. [6,16,32] means 6 filters are used in the first conv layer, 16 filters are used in the second conv layer and finally 32 filters are used in the third conv layer. The result is discussed in part 4.

4. The highest accuracy is 0.76171875 with the number of layers 3
   The average accuracy of 1 layer is 0.5724609375
   The average accuracy of 2 layers is 0.6234375
   The average accuracy of 3 layers is 0.672265625
   Compare to the previous experiment, the Lenet 5 increase the accuracy up to 5% on average. The best result is boots up to 76% which is nice. Along with that, the highest accuracy on Lenet just need to run 8 epochs, on the other hand, the traditional one has run up to 20 epochs to achieve the maximum. Overall, we Lenet model is considered to be a good model in solving the current problem.
5. Here are the plots of the images found in layer 1, 2 and 3 respectively after the LeNet5 model - network has been trained by 20 epochs:

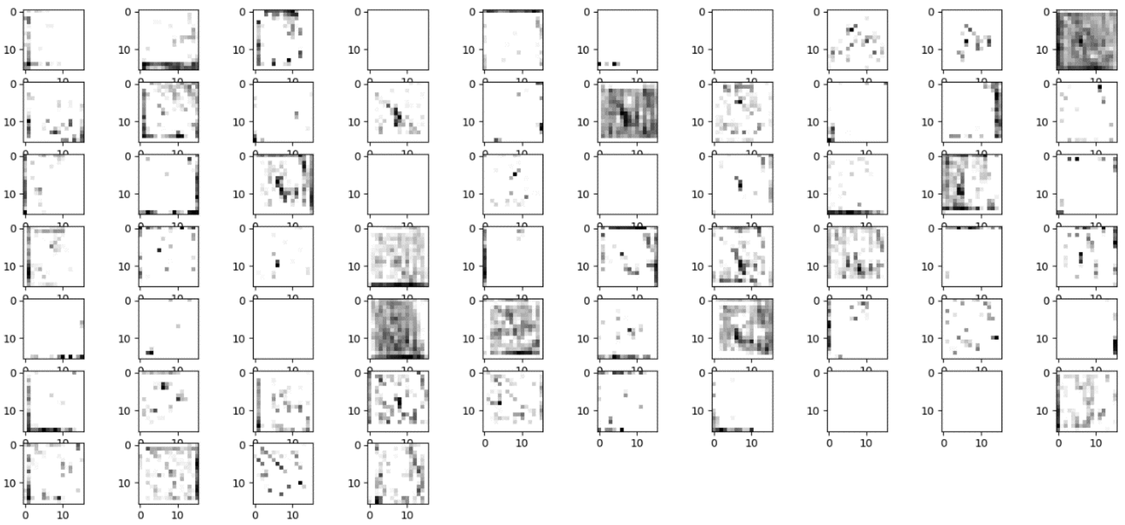


Original image

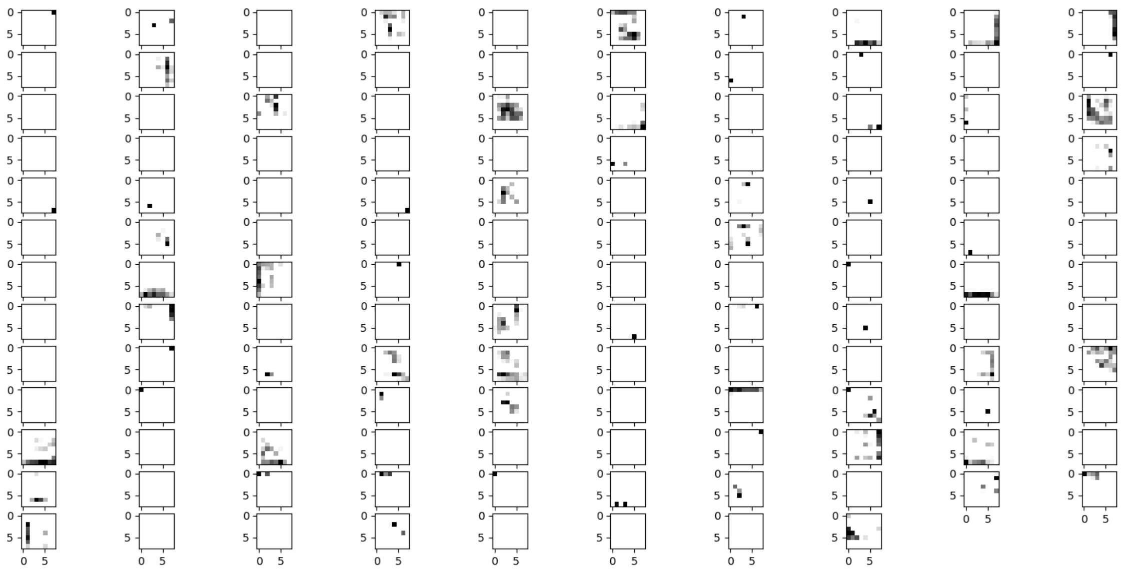
The first layer:



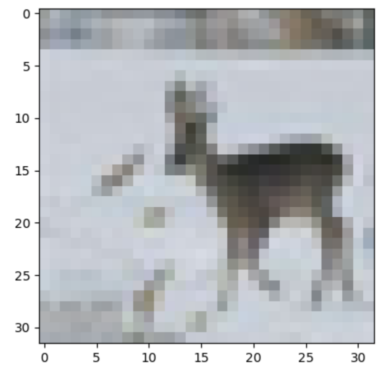
The second layer:



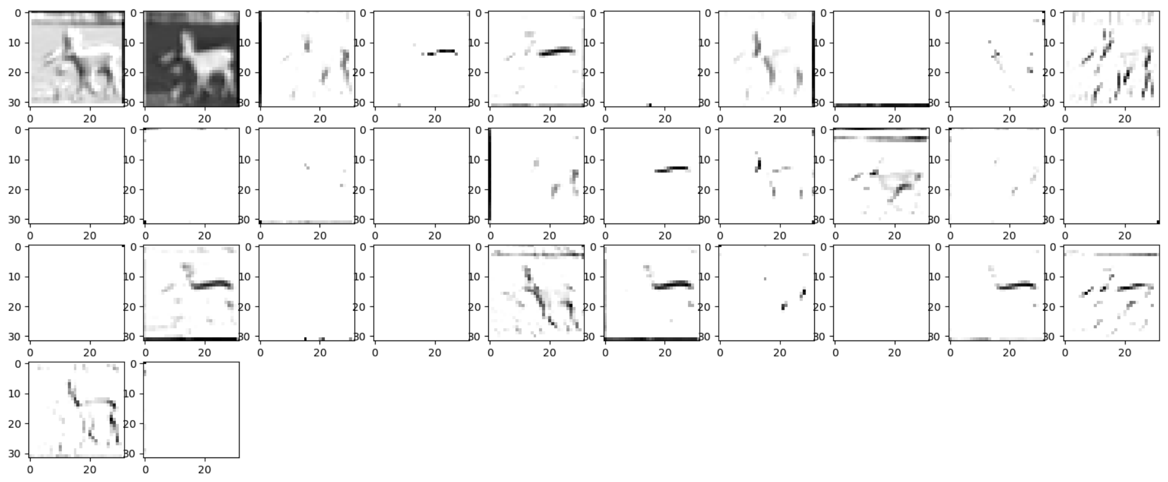
The third layer:



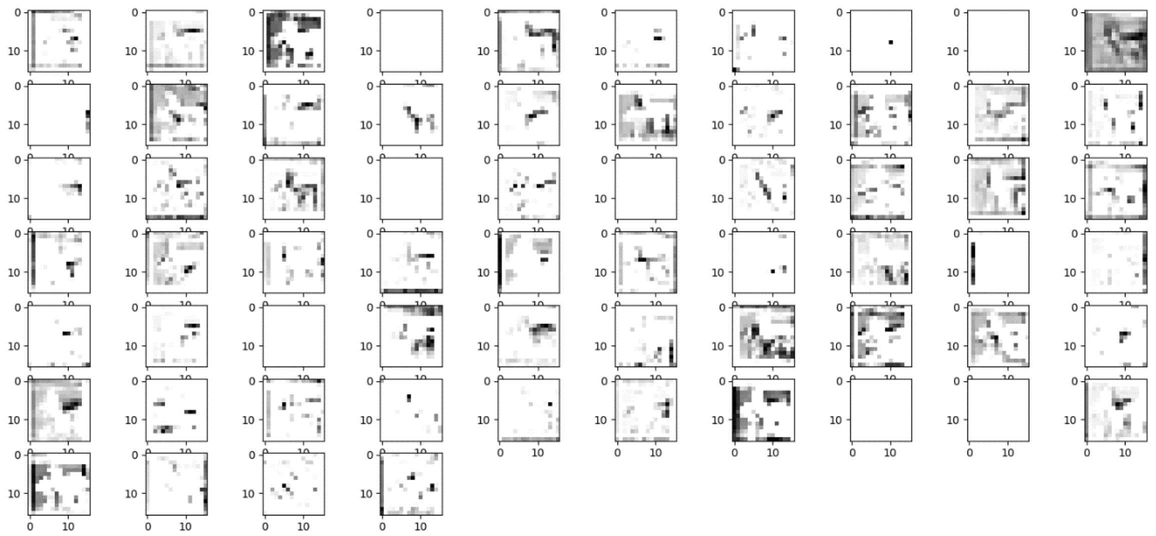
The original image:



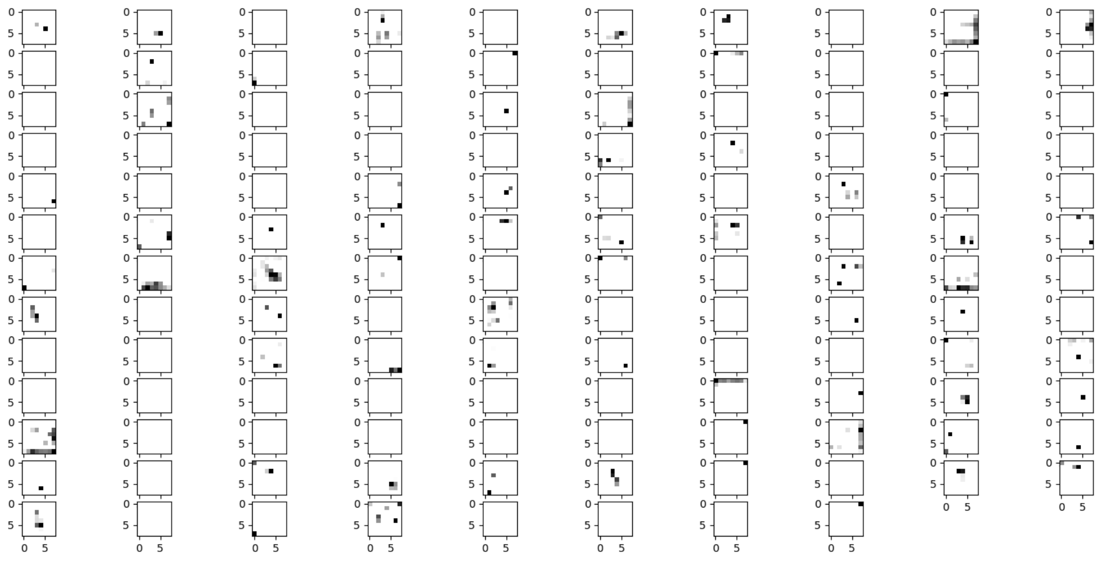
Its first layer:



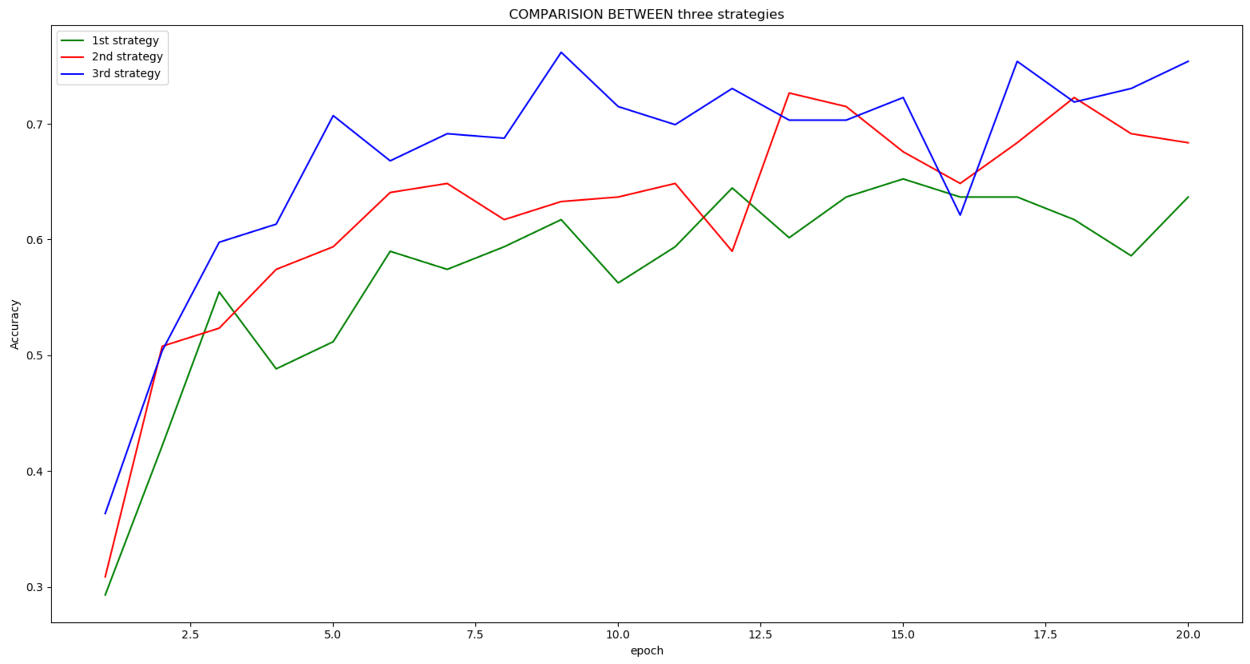
Its second layer:



Its third layer:



6. Here is the graph of the second experiment, which is the competition of filter sets. The winner in the experiment is: [32,64,128]



7. Here is the table of the accuracy:

[6,16,32]	[16,32,64]	[32,64,128]
0 0.29296875	0 0.30859375	0 0.36328125
1 0.421875	1 0.5078125	1 0.50390625
2 0.5546875	2 0.5234375	2 0.59765625
3 0.48828125	3 0.57421875	3 0.61328125
4 0.51171875	4 0.59375	4 0.70703125
5 0.58984375	5 0.640625	5 0.66796875
6 0.57421875	6 0.6484375	6 0.69140625
7 0.59375	7 0.6171875	7 0.6875
8 0.6171875	8 0.6328125	**8 0.76171875**
9 0.5625	9 0.63671875	9 0.71484375
10 0.59375	10 0.6484375	10 0.69921875
11 0.64453125	11 0.58984375	11 0.73046875
12 0.6015625	**12 0.7265625**	12 0.703125
13 0.63671875	13 0.71484375	13 0.703125
14 0.65234375	14 0.67578125	14 0.72265625
15 0.63671875	15 0.6484375	15 0.62109375
16 0.63671875	16 0.68359375	16 0.75390625
17 0.6171875	17 0.72265625	17 0.71875
18 0.5859375	18 0.69140625	18 0.73046875
19 0.63671875	19 0.68359375	19 0.75390625