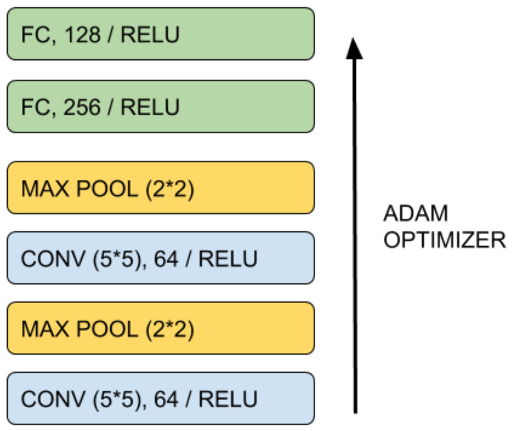
ASSIGNMENT- 1

By Neha (IMT2014018) & Meghana (IMT2014034)

Aim: To fine tune moderately deep CNN so that it fits well to CIFAR-10 dataset

Dataset: CIFAR-10 dataset has 60000 images of 10 object categories. It has 50000 images for training and 10000 images for testing.

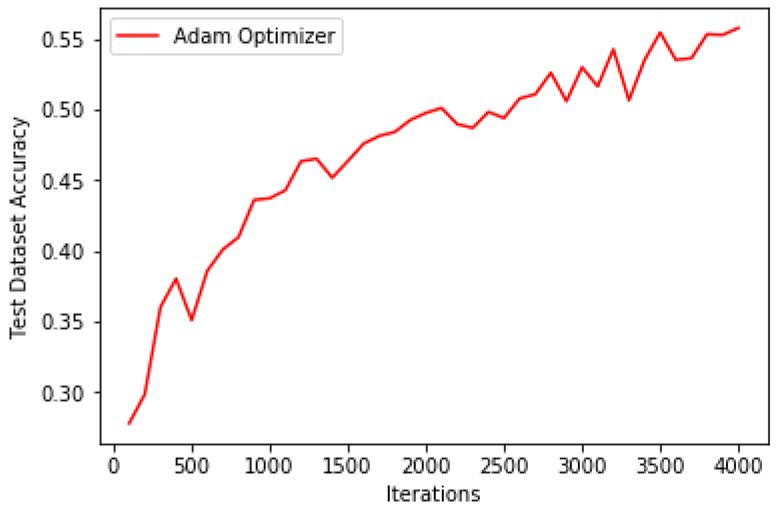
Initial Architecture: The neural network has 2 convolution layers and 2 fully connected layers, followed by a softmax layer. The first convolution layer has 64 filters each of size 5*5. It is followed by a max pooling layer of size 2*2. The second convolution layer also has 64 filters each of size 5*5. This layer is also followed by a max pool layer. The size of the first fully connected layer is 256 and followed by another fully connected layer of size 128.



Preprocessing: We have preprocessed images in the training set by randomly changing the hue, brightness and contrast of an image. These preprocessed images are later used to train the neural network.

Results:

1. Effect of number of iterations on test dataset accuracy:



As expected, as the number of iterations increased, performance of the net also increased. We capped the number of iterations at 4000 because of the time it was taking for training.

Test Dataset Accuracy: 55.8%
Training Time*: 24 mins

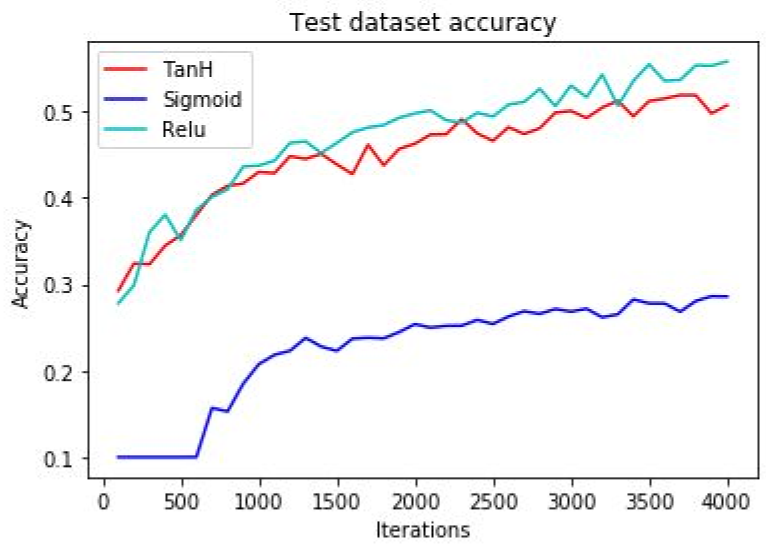
Please note that the training time is high because we are calculating accuracy on entire test dataset after every 100 iterations (to visualize above plots) which adds to the training time, otherwise training time decreases considerably.

For the discussion below training time with * above includes testing time after every 100 iterations. Because there is relative comparison between different available options, we expect this won't be a problem. Also, the number of iterations have been fixed to 4000.

2. Comparison of performance using different activation functions:

   Used ADAM optimizer and same activation function for all the layers.

Activation function	Training time*	Final Accuracy
ReLu	**29 mins 44 secs**	**55.8%**
TanH	35 mins 23 secs	50.7%
Sigmoid	36 mins 13 secs	28.6%



Observation: It is clearly seen that ReLu activation function takes less time to train and also gives a better performance compared to other activation functions, namely sigmoid and tanh.

3. Effect of batch normalization(BN) on test dataset accuracy:

   Used ADAM optimizer with ReLu as activation function for all the layers.

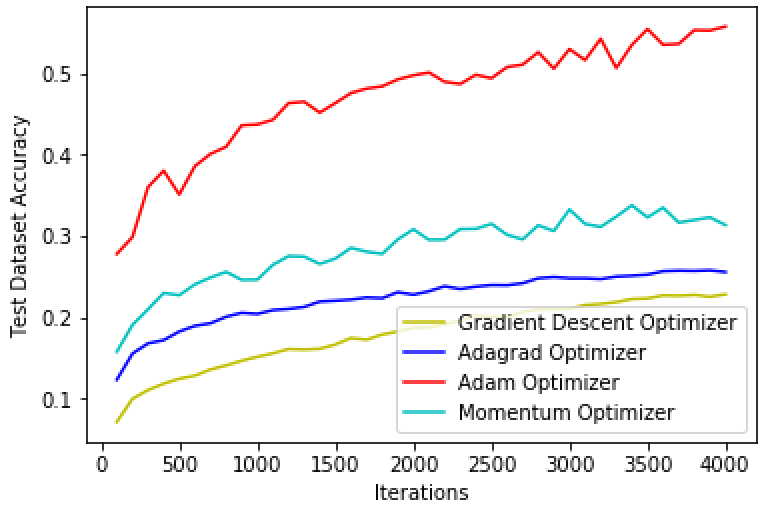
	Training time	Accuracy
No batch normalization	29 mins 44 secs	55.8%
Applying BN only to first convolution layer	**34 mins 40 secs**	**57.2%**
Applying BN only to second convolution layer	26 mins 30 secs	55.4%
Applying BN to both the convolution layers	29 mins 28 secs	55.5%

   Observation: Neural net’s performance improves when batch normalization is applied to first convolutional layer but doing the same to all the convolutional layers has not really affected the accuracy (slightly decreased).

4. Effect of changing the optimizer on test dataset accuracy:

Used ReLu as activation function for all the layers.

Optimizer	Training time*	Accuracy
Gradient Descent Optimizer	34 mins 30 secs	22.8%
Adam Optimizer	**29 mins 44 secs**	**55.8%**
Momentum Optimizer	28 mins 25 secs	31.3%
AdaGrad Optimizer	1 hr 6 min 24 secs	25.6%

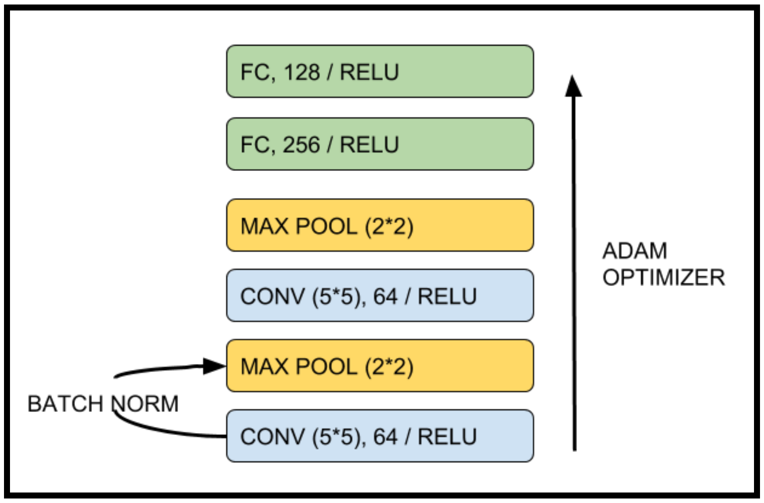


Observation: We can see from the plot that using moment or other adaptive learning optimizers give better results than using plain Gradient Descent. Also, Adam optimizer improves accuracy quite significantly as compared to all the other optimizers and in fact takes less time than Gradient Descent or Adagrad.

<u>Recommended Architecture</u>:

Based on above experiment, we recommend the following architecture:

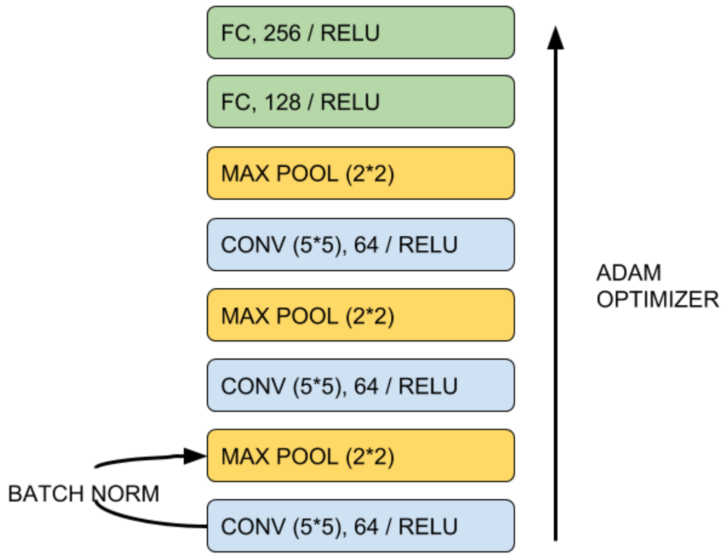
FINAL RECOMMENDED ARCHITECTURE



Training Time: 34 min 40 secs
Test dataset accuracy: 57.2%

We have submitted code corresponding to the above recommended architecture.

Just to see the effect of depth, we added another layer to our recommended architecture as follows:



Training Time: 52 mins 22 secs
Test dataset accuracy: 55.5%

Observation: Unexpectedly, accuracy decreased when we added another convolutional layer. We suspect that adding more layers would increase the performance but due to time and computational constraints, we haven’t played with more number of layers.