1. **Convolutional layer: Write code that instantiates a 5-by-5 kernel of 32 filters for the image, and perform a 2d convolution of the image with stride 1 in each direction. Define a bias variable of shape [32] for the output of the kernel, and add the bias to the output of the 2d convolution. What will be the output tensor dimensions if we had used 64 filters of 5-by-5 and a stride of 1? What will it be if we used 32 filters of 7-by-7 and a stride of 2?**

**Output tensor Dimension with 64 filters of 5-by-5 and a stride of 1:**

Let’s define

O = Size (width) of output image.  
I = Size (width) of input image.  
K = Size (width) of kernels used in the Conv Layer.  
N = Number of kernels.  
S = Stride of the convolution operation.  
P = Padding.

The size (O) of the output tensor is given by

\[ O = \frac{I - K + 2P}{S} + 1 \]

The number of channels in the output image is equal to the number of kernels.

In our case,  
I = 32  
K = 5  
N = 64  
S = 1  
P = 1

Therefore O=(32-5+2)/1+1=30

Number of channels=64

So the tensor matrix= 64 x [30]

**What will it be if we used 32 filters of 7-by-7 and a stride of 2?**

Therefore O= (32-7+2)/2+1=18.5 ~ 19 (Automatic Round off by Tensorflow)

Number of channels=32

So the tensor matrix= 32 x [19]

**4. Learning: Use cross entropy as the loss, as you’ve done before, and train the model. Plot the training and validation loss, as well as the training and validation accuracy over 50 epochs of training. Run on 3 different hyperparameter settings (i.e. change the learning rate, weight decay coefficient, dropout) and report your results.**

*The accuracy results for all parameters are as follows:*

*##### MODEL PARAMETERS-5 learning\_rate = 0.001 num\_steps = 2000 batch\_size = 32 #####Testing Accuracy: 0.15053764*

*##### MODEL PARAMETERS-1 learning\_rate = 0.001 num\_steps = 2000 batch\_size = 32 #####Testing Accuracy: {'accuracy': 0.16129032, 'loss': 1.8621702, 'global\_step': 0}*

*##### MODEL PARAMETERS-2 learning\_rate = 0.0001 num\_steps = 5000 batch\_size = 32 #####Testing Accuracy: {'accuracy': 0.17204301, 'loss': 1.8332964, 'global\_step': 0}*

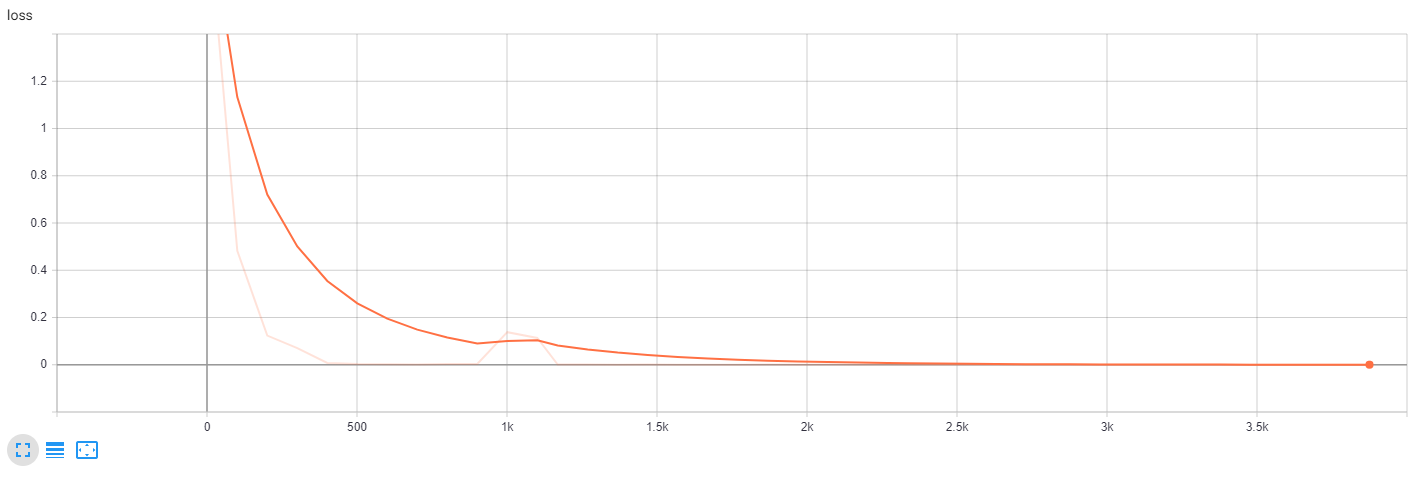
*##### MODEL PARAMETERS-3 learning\_rate = 0.001 num\_steps = 2000 batch\_size = 128 #####Testing Accuracy: {'accuracy': 0.16129032, 'loss': 1.8754494, 'global\_step': 0}*

*##### MODEL PARAMETERS-4 learning\_rate = 0.001 num\_steps = 1000 batch\_size = 256 #####Testing Accuracy: {'accuracy': 0.19354838, 'loss': 1.827026, 'global\_step': 0}##### MODEL*

*PARAMETERS-5 learning\_rate = 0.001 num\_steps = 2000 batch\_size = 32 #####Testing Accuracy: {'accuracy': 0.15053764, 'loss': 1.8067622, 'global\_step': 0}*

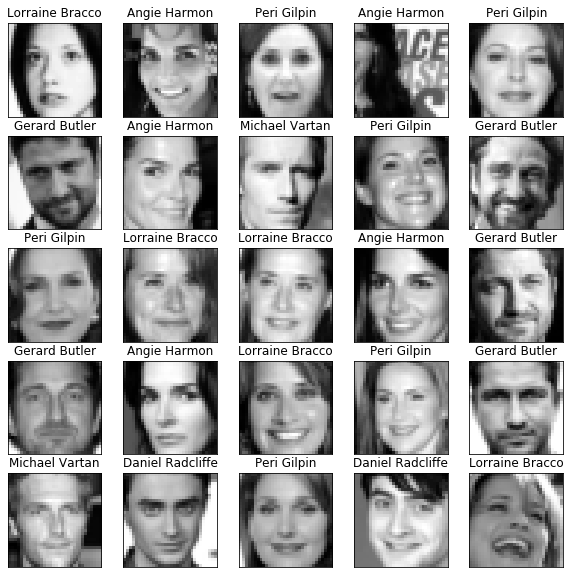
It is clear from the above results that no matter what, with only one conv layer, the test accuracy never increases beyond 20%. More conv layer is thus required. Though it is also evident that the dropout layer did its part in restricting overfitting.

*The average loss during training for all parameter changes is as follows:*

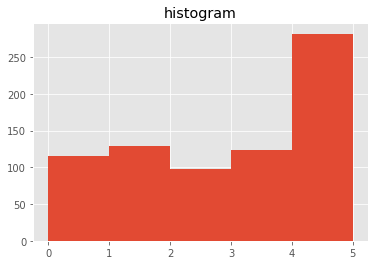


**5. Visualization: To get more insight into what a convolutional neural network achieves with its architecture, you will visualize the function that the convolutional layer provides. Visu-alize 8 of the 5-by-5 kernels trained in question 1, comment on what the network is trained to recognize with these kernels and how further layers of convolution may improve the performance.**

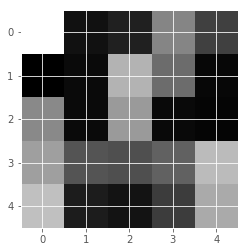
**Images:**



**Histogram:**

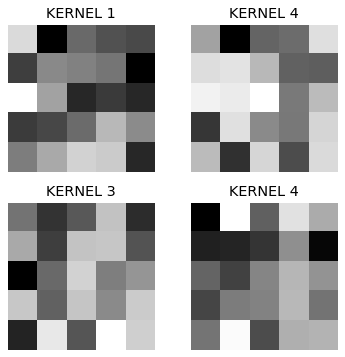


**Kernels:**



**Kernel 1: This kernel somehow manages to create a filter which finds the outline of the face.**

**Let’s investigate on the other kernels:**



**I could not any distinct features which these filters are finding. Maybe a different visualization technique would be beneficial.**