Same approaches has been used for the two tasks, the difference is only in the number of classes which is 6 for the face recognition and 2 for the Gender recognition.

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?

   Code implemented

   We count the number of time a filter will be applied to one dimension (based on the stride used and the padding). In order to do so we may use this formula : $(N-F)/S+1$

   N : dimention of image

   F: dimetion of the filter

   S: Strides

   ((32 – 5) / 1)+ 1 = 27 + 1 = 28

   28 x 28 x 64

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

   ((32-7) / 2) + 1 = depending on the assumption of non-integer result = 13 or 14

   13 x 13 x 32 or 14 x 14 x 32 depending on the padding used

2. Max pooling layer: Write code that instantiates a max pooling layer of size 3-by-3 and a stride of 2 in each direction.

```
model.add(MaxPooling2D(pool_size = (3, 3), strides = 2))
```

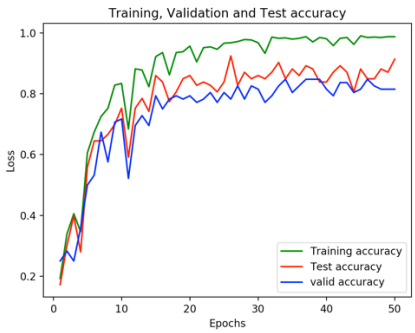
Please refer to Code

3. Fully connected layer: Now flatten the output of the previous layer and pass through two hidden layers with ReLU activations of size 384 and 192 with a dropout rate of 0.5. At the final layer, output the probabilities of predicting each celebrity ID

```
model.add(Flatten())

model.add(Dense(units = 384, activation = 'relu'))

model.add(Dropout(0.5))

model.add(Dense(units = 192, activation = 'relu'))

model.add(Dense(units = num_classes, activation = 'softmax'))
```

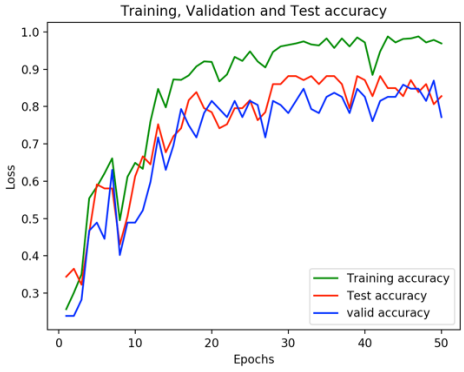
4. Learning: Use cross entropy as the loss, as you've done before, and train the model. Plot the training and validation loss, as well as the training and validation accuracy over 50 epochs of training. Run on 3 different hyperparameter settings (i.e. change the learning rate, weight decay coefficient,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

Dropout 0.5 Weight decay 0.01 Learning rate 0.001



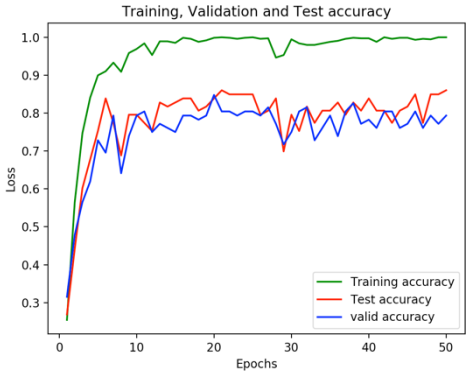
In this experiment we were able to correctly classify 85/93 validation data with loss of 0.3673 and accuracy of 89.24 %

Dropout 0.5 Weight decay 0.001 Learning rate 0.001



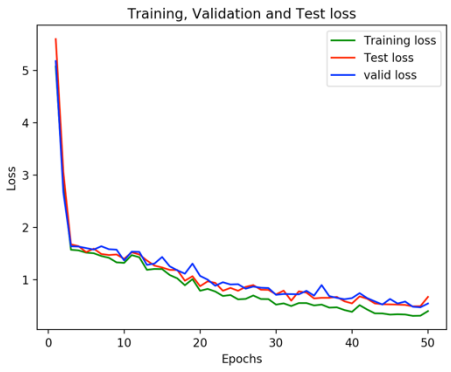
In this experiment we were able to correctly classify 77/93 validation data with test loss of 0.4814 and test accuracy of 82.79 %

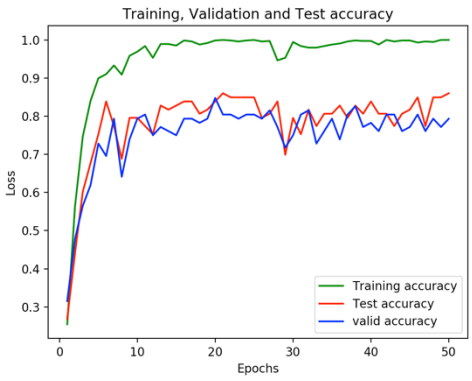
Dropout 0.5 Weight decay 0.01 Learning rate 0.002



In this experiment we were able to correctly classify 80/93 validation data with test loss of 0.6468 and test accuracy of 86.02 %

Dropout 0.4 Weight decay 0.01 Learning rate 0.0001





In this experiment we were able to correctly classify 69/93 validation data with test loss of 0.6703 and test accuracy of 74.19 %

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. Visualize 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.

   By increasing or decreasing the number of layers and or number of nodes in each convolutional layer we can improve the performance.
   We can also improve performance by tuning the hyper parameters.
   Preproccessing the image may also have effect on the performance.

Here we have presented the 8 of 5x5 kernels trained weights from the second last layer.
It's worth noting that the layers that are deeper will show more specific features of the images, however, the very first layers are usually responsible for distinguishing the background, the edges, lines and etc.