Question 2

(a) Downloading the data.

(b) Data Filtering for birds:

To begin with I used the cifar10 load_data method. But this method loads all the examples.

Looking at the meta data text file I was able to determine the index of the images in category ‘birds’.
And then picked those whose among the whole data whose indices matched with that of birds.

Here is a snippet:

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

X_train, y_train = X_train[y_train == Bird_label], y_train[y_train ==
Bird_label]
```

(c) Picking pixels:

I decided to pick 40% of the pixels.

(d) Determining the cluster centers for k=4

After running the k-means algorithm on the data at hand, these were the centers I obtained:

```
Cluster Centers after k-means, k=4:

[ [156.31445673, 155.69876765, 134.712665]
[49.06789563, 51.0214536, 37.59833464]
[206.62222261, 211.86548477, 210.5235235]
[108.02356496, 108.70112973, 83.86542385] ]
```

(e) Image shape Transformation

I used the transofrm from skimage to reshape the image to 32 x 32 x 1.

Here is the snippet of how it works:

```
from skimage import transform
new_features = np.array(list
                        (map
                         (lambda img: transform.resize(
```

```
                                        img.reshape(32,32,3),#old shape
                                          (32, 32,1), #new shape
                                          mode='constant',

                                         #flatten the resized image

                                          preserve_range=True).ravel(),
            features)))
```

(f) The CNN Model:

Approach:

- At first I used my PC to run 6 epochs and compare the results for the asignment. I then ran it on a supercomputer with TESLA- K80 GPU for tensorflow and was able to run high amount of Epochs for this test, out of curiosity.

- I decided not to use the padding and used a stride of length two.

- Input is of shape (32, 32) which is a Greyscale Image.
- Using the cluster centers obtained I recoloured images(having only 4 colours) as the output and not the original CIFAR-10 images.
- The output is expected to have a shape (1024, 4) where 1024 is the number of pixels and 4 is the one hot encoded value of that pixel. In other words, for instancce pixel 1 has colour 2, [0, 0, 1, 0].
- Moving forward, the second layer in the network will have 4096 units, which will then be reshaped to (1024, 4) units using another Reshape layer.
- After this, I obtained a 2D grid in the network of the shape (1024, 4) (which matches the expected output shape). I the ndecided to add another layer , namely the Activation layer with softmax which helped to choose the highest value out of 4 inputs for each of the 1024 pixels.
- After recieiving the 1024 output counts I built a compilation for the model with loss='categorical_crossentropy'.

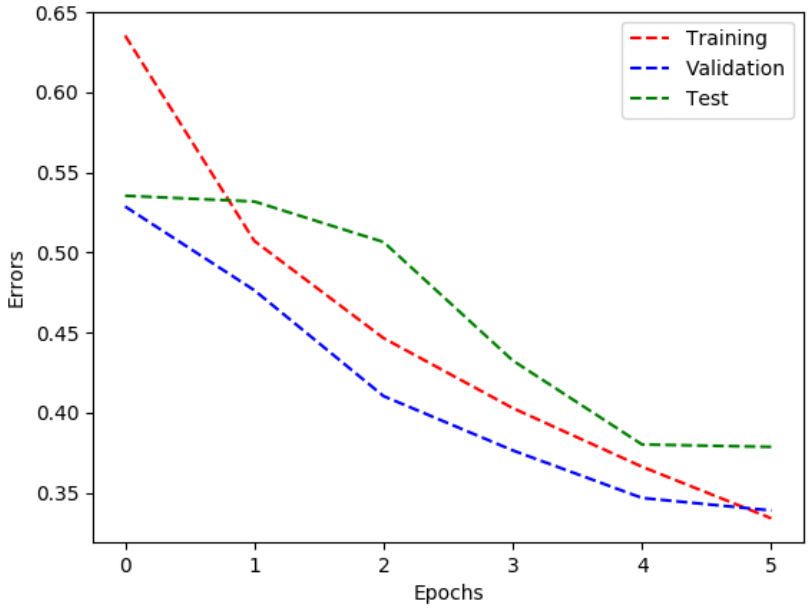
The validation set of chosen as 500 of the bird samples, training as 4500 and the test set had 1000 instances.

Accuracy:

```
4500/4500
Standard Training.
Train on 4500 samples, validate on 500 samples
Epoch 1/6
4500/4500 [==============================] - 34s - loss: 1.7511 - acc: 0.3645 -
val_loss: 1.4763 - val_acc: 0.4601
Epoch 2/6
```

```
4500/4500 [==============================] - 34s - loss: 1.4102 - acc: 0.4927 -
val_loss: 1.2470 - val_acc: 0.5588
Epoch 3/6
4500/4500 [==============================] - 34s - loss: 1.2466 - acc: 0.5534 -
val_loss: 1.1544 - val_acc: 0.5939
Epoch 4/6
4500/4500 [==============================] - 34s - loss: 1.1335 - acc: 0.5971 -
val_loss: 1.1024 - val_acc: 0.6102
Epoch 5/6
4500/4500 [==============================] - 34s - loss: 1.0310 - acc: 0.6337 -
val_loss: 1.0466 - val_acc: 0.6321
Epoch 6/6
4500/4500 [==============================] - 34s - loss: 0.9422 - acc: 0.6659 -
val_loss: 1.0070 - val_acc: 0.6489
```

After Running for 6 epochs the result obtained is :



After running the model on the supercomputer:

```
4500/4500
Standard Training.
Train on 4500 samples, validate on 500 samples
Epoch 1/100
4500/4500 [==============================] - 34s - loss: 1.7511 - acc: 0.5105 -
val_loss: 1.4763 - val_acc: 0.4601
Epoch 2/100
4500/4500 [==============================] - 34s - loss: 1.4102 - acc: 0.5127 -
val_loss: 1.2470 - val_acc: 0.5588
.
.
```

.

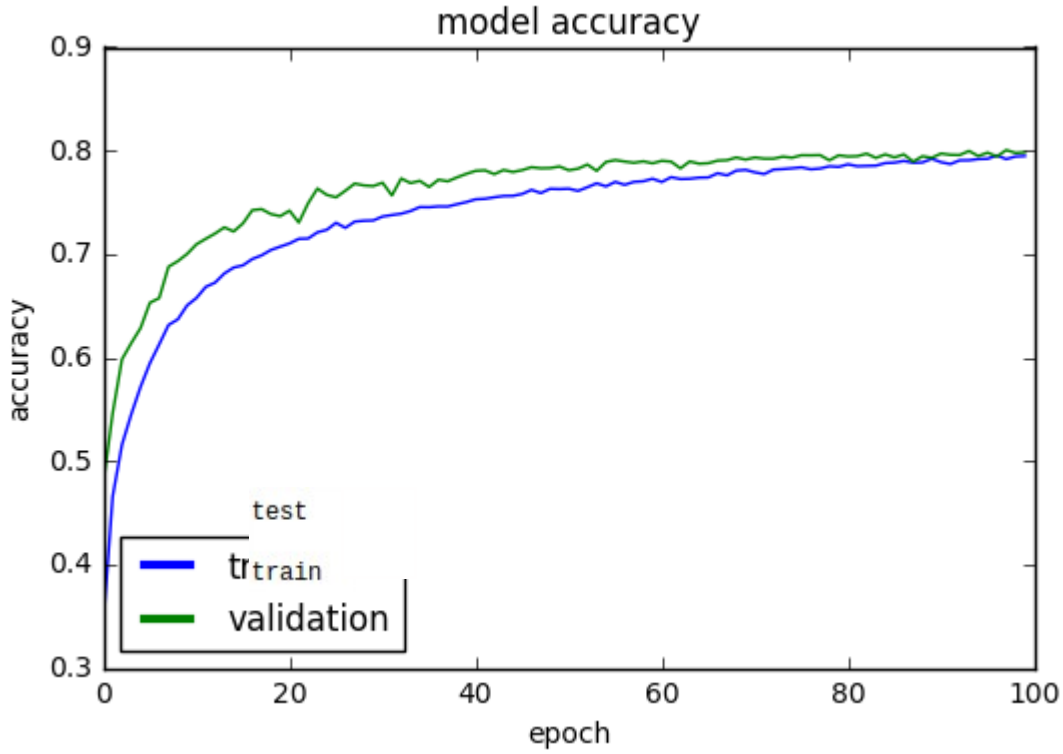
```
Epoch 96/100
4500/4500 [==============================] - 34s - loss: 0.0958 - acc: 0.7598 -
val_loss: 1.4032 - val_acc: 0.7032
Epoch 97/100
4500/4500 [==============================] - 34s - loss: 0.0896 - acc: 0.7599 -
val_loss: 1.4049 - val_acc: 0.7035
Epoch 98/100
4500/4500 [==============================] - 34s - loss: 0.0857 - acc: 0.7600 -
val_loss: 1.3944 - val_acc: 0.7009
Epoch 99/100
4500/4500 [==============================] - 34s - loss: 0.0889 - acc: 0.7603 -
val_loss: 1.4177 - val_acc: 0.7050
Epoch 100/100
4500/4500 [==============================] - 34s - loss: 0.0870 - acc: 0.7652 -
val_loss: 1.4261 - val_acc: 0.7014
```

Prediting Test Set:

```
y_pred = model1.predict_classes(X_test)
 1000/1000 [==============================>.] - ETA: 0s

Number of true predictions:  765
Number of false predictions: 235
```

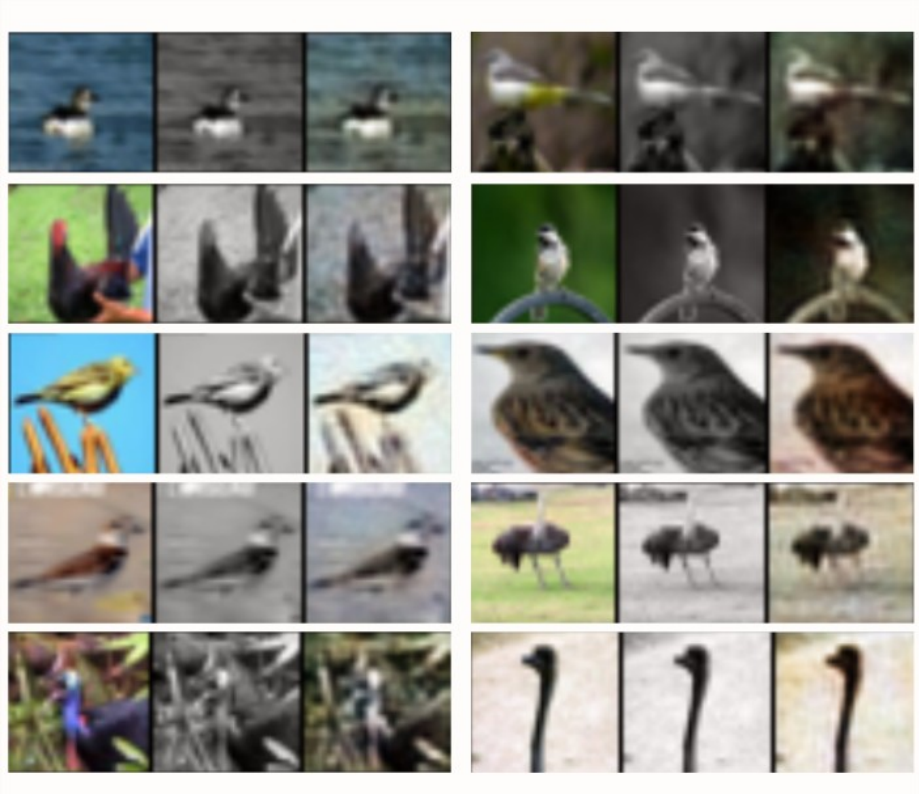
Comparison of Train and Test Set Accuracy:



Colors predicted by the k-means clustering:

I then observed, for the first ten images of birds, the actual image, grayscale version and the opredicted output by the CNN, in that order.

Following were the results obtanied:



Clearly, the CNN struggles to predic tteh hues of bluw and green. To put it simply the vibrant and bright colours.
However, it does a more than a decentjob in colouring subtle colors and darker shades.